Tracing Centre-lines of Digital Patterns using Maximal-Square and Maximal-Circle Algorithms

Mabrouk R. Abairid

A

Major Technical Report

in

The Department

of

Computer Science

Presented in Partial Fulfillment of the Requirements
for the Degree of Master of Computer Science at
Concordia University
Montréal, Québec, Canada

May 1987

ISBN 0-315-35557-3

Abstract

Tracing Centre-Lines
of Digital Patterns
using
Maximal-Square and Maximal-Circle
Algorithms

Mabrouk R. Abairid

In this report we compare two thinning algorithms which produce the centre-lines of digitized binary patterns. The first algorithm is based on maximal squares (MS), while the second is based on maximal circles (MC). Each algorithm consists of two versions: In version 1 (MS1 and MC1)] of the MS and MC algorithms, the maximal squares/circles are made up of black pixels only; whereas in version 2 (MS2 and MC2), a maximal square/circle may include a few white pixels.

To save memory space, the binary patterns are stored in a link-list structure instead of matrix. Since MS and MC algorithms tend to produce discontinuities and noisy skeletons, a filling and smoothing algorithm has been added to correct this defect.

These algorithms are compared experimentally with those of other research groups. The results show that the modified MS and MC algorithms produce centre-lines with good to excellent quality.

ACKNOWLEDGEMENTS

I like to thank my supervisor Prof. C. Y. Suen, for his support and encouragement. His help, suggestions, advice and criticisms have been the key to the successful results which were achieved in this report. I also wish to thank my wife and my sons Yousif, Abdelbast, Ziad and Tariq for their moral support.

This research project has been supported by the Natural Sciences and Engineering Research Council of Canada.

Table of Contents

1 Introduction

Thinning is a process by which a binary pattern is transformed into another one consisting of its centre-line. The idea was first introduced about three decades ago [2]. Its applications are numerous, e.g. Classification of fingerprints [4]; the inspection of printed circuit boards [5]; the recognition of characters [3, 6]; analysis of chromosome shapes [7]; counting of asbestos fibers on air filters [8], to name a few. The major motivations behind thinning in pattern recognition and image processing are :

(1) To reduce data storage.
(2) To reduce transmission requirments.
(3) To pare down the amount of data to be processed.
(4) To facilitate the extraction of features from the pattern.

Over the last two decades, numerous thinning algorithms have been developed [1, 2, 3, 7, 9, 10]. The predominant problems experienced by most of those algorithms are :

(a) Maintaining connectedness.
(b) Retaining end-points.
(c) Ensuring points are removed symmetrically.

Most existing algorithms produce centre-lines by successive stripping of the contour points of the binary pattern. Due to the above mentioned problems, different degrees of distortion have also been produced. The objective of our work is to develop a thinning algorithm which produces smooth centre-lines with the ability of reconstructing the original pattern with the least amount of distortion.

The Maximal squares (MS) algorithm was first proposed by Wakayama [1]. In this algorithm, the centre-line was derived from all centres of the maximal squares moving across the binary pattern. In MS algorithm, the centre-lines are developed by defining initial squares on the objects (pixels), and by enlarging them until they cannot be enlarged any further within the binary pattern. These squares are called maximal squares. The MS algorithm consists of two versions : MS1 and MS2. MS1 is an implementation of Wakayama's algorithm, whereas MS2 is a modification of MS1. The difference will be explained in the next few pages.

We also introduce the Maximal-Circles (MC) algorithm where the centre-lines are derived in a manner similar to those obtained from the MS algorithm, i.e. instead of squares we use circles. The MC is also consists of two versions : MC1 and MC2.

MS1,MC1 :

In these two algorithms the centre-lines are produced by the centres of all maximal squares/circles of the binary pattern. A maximal square/circle consists of black pixels (value 1) only.

MS2,MC2 :

These two algorithms perform in such a way that, along with black pixels a maximal square/circle may contain a small portion of white ones; and the number of white pixels is a function of the size of the square/circle.

The inclusion of a few white pixels on the periphery of the square/circle in MS2 and MC2, offers the flexibility of accommodating a greater variety of pattern shapes.

2 Definitions

A binary digitized picture is reresented by a matrix MAT(K,N), where each pixel MAT(J,P) is either 0 or 1 (Fig. 1-a). In both the MS and MC algorithms, only those pixels with a value of 1 are processed, pixels having a 0 value are not.

Binary patterns consist of a number of rows, and in turn each row may also consist of one or more groups. A group is a set of consecutive (connected) black pixels in the same row. The (m)th group G in (J)th row is defined as :

$$G_J^m = \{ Mat(J,P) \mid \quad \forall \quad Mat(J,P) = 1 \}$$

$$\text{Where} \quad P = B_J^m, (B+1)_J^m, \ldots\ldots, E_J^m$$

B is the beginning-point of the (m)th group
E is the ending point of the (m)th group.

In Fig. 1-a, the entire pattern consists of 9 connected groups. In row 1 for example, there is only 1 group which starts at column 6 and ends at column 8 (3 pixels). Row 2 has 2 groups, the first of which has two pixels and starts and ends at columns 1 and 2 respectively.

Instead of the conventional way of storing a pattern in a matrix, we store it in a Link-List structure descriptive representation (Fig. 1-b). To create this structure, the pattern has to be read (from a digitizer, or from memory) in a row-by-row fashion. Once the pattern has been read, each group's beginning (B) and ending (E) points were saved in a link-list data structure (Fig. 1-b).

The relationship between adjacent groups in row J is maintained by linking the first group to the second, second to the third and so on. A second relationship between a group in row J and a group in row J+1 is also maintained by linking (Fig. 1-b).

$$\text{Rule (I)}: \{(B^{m}_{J+1} \le E^{n}_{J}) \text{ and } (E^{m}_{J+1} \ge B^{n}_{J})\}$$

Where J represents the row number, and
m, n are the group indices.

The second relationship is subject to Rule (1), which states that two groups m, n in two consecutive rows (J, J+1) must be connected to each other at least at one point (pixel). If more than one group satisfy rule (1), then the (n)th group of the (J)th row refers only to the first. Fig. 1-b, for example, illustrates that the 2nd group in the 2nd row is linked to the 1st group in the 3rd row.

Rows	Columns: 1	2	3	4	5	6	7	8	9	10	11
1	0	0	0	0	0	1	1	1	0	0	0
2	1	1	0	0	1	1	1	1	1	1	1
3	1	1	1	1	1	1	1	1	0	1	1
4	1	1	1	1	1	0	1	1	1	0	0
5	0	1	1	1	0	0	0	1	1	0	0

Fig. 1-a

A binary pattern composed of five rows. The 1st row has one group while rows 2-5 have two groups each.

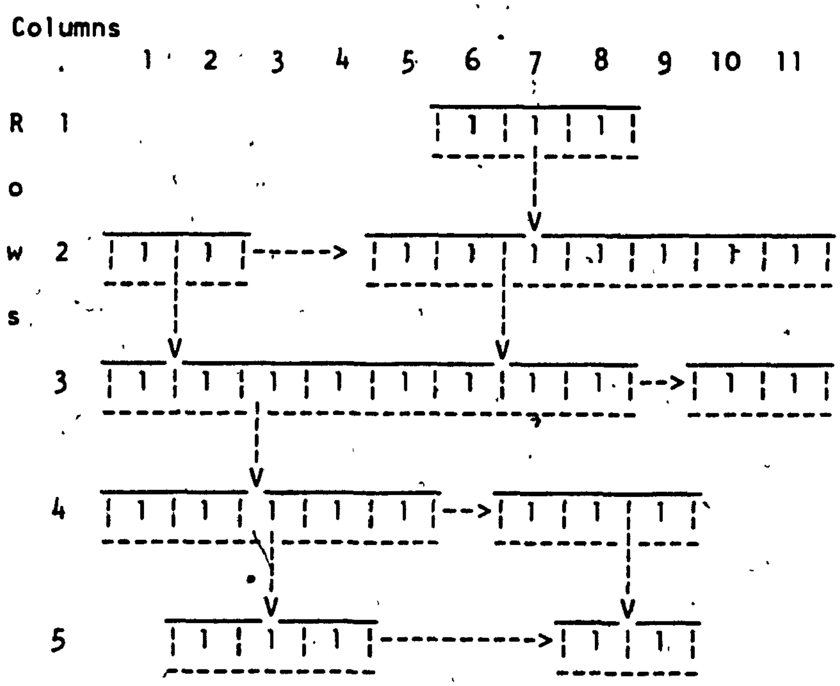


Fig. 1-b

Structural representation of the binary pattern of Fig. 1-a.

Fig. 1

Referring to Fig. 1, the only group in row 1 is linked to the second group of row 2, while the second group of row 2 is linked to the first group of row 3. Also the first groups of rows 2-5 are all linked to their second groups respectively.

3 Maximal Square/Circle Algorithms

In the proposed maximal square/circle algorithms for thinning binary patterns, the centre-lines are defined by all centres of maximal squares/circles enlarged on the pixels of each group in the pattern. In order to define a square/circle, we utilize the link-list which contains the beginning and ending (B and E) of each group in the pattern.

3.1- Outline of the algorithms

<u>Step-1</u> : Define a square/circle

Propagating through all groups in the pattern (left to right, top to bottom), a pixel at a time, a square/circle consisting of one pixel (at Mat(J, P)) is first defined (see Fig. 2).

<u>Step-2</u> : Enlarge the square/circle

To enlarge this square/circle of size 1(L); the B and E of the current group in the current row J are used together with a group in the (J+L)th row which satisfies rule (1) and one of the following two new rules. Rule (2) is applied by the MS algorithm for maximal squares and rule (3) is applied by the MC algorithm for maximal circles.

Rule (2)

a : $\{(B^{m}_{J+L} \le P^{n}_{J}) \text{ and } (E^{m}_{J+L} > P^{n}_{J} + L)\}$

b : The pixel at position (P+L) in all groups comprising a size (L) square must be black.

Rule (3) : $\{(B^{m}_{J+L} <= P^{n}_{J}-C)$ and $(E^{m}_{J+L} >= E^{n}_{J}+C)\}$

Where C ---> Fixed value for each group in the circle
J ---> Current row.
L ---> Current circle size.
m,n ---> Group indices in rows J and J+L
P = B, B+1, ..., E-1, E.

Rule (2) states that the (m)th group on (J+L)th row must be at least one pixel longer than L (size of the square) in order to enlarge the size (L) square to (L+1). Whereas rule (3) requires that, the (m)th group in (J+L)th row passes through a potentially larger circle (L+1), and the interval (B-C to E+C) lies inside this circle. For all possible maximal circles (in our experiment we have sizes 1 to 15), the constant 'C' in rule (3) was predetermined (Table 1) and used by both MC1 and MC2 algorithms.

If rule (1) plus either rule 2 or rule 3 hold, then L is incremented by 1, and another group in row J+L is tested by rules (1), (2) and (3) for a possible bigger square/circle. The iteration process of enlarging a square/circle (step 2) continues until one or more rules fail. At this point, L also represents the number of groups comprising this square/circle. If a square/circle cannot be enlarged further, step 3 is invoked.

Step-3 : Check for duplicate or redundant square/circle

It compares this newly defined and enlarged maximal square/circle to all squares/circles created previously to see whether or not it is a duplicate, or a most likely situation, that is, it is inside a larger one. If the test proves to be negative or this new square/circle is the first one to be created on the pattern, add this square/circle to the maximal square/circle list.

Step-4 : Propagate through current group/jump to next row

Advance to the next pixel of the current group and repeat the steps 1-3 unless one of the following conditions is true :

Maximal square condition :

the ending point (E) of the first group in the square is less than or equal to (R+L).

Maximal circle condition :

For all groups passing through the maximal circle $\{ (E_{R+i} \le E_C) \}$ is true.

where : R --> is 1st group of the circle.
i --> 0, 1, ..., L (circle size).
C --> group passing through circle centre.

In other words, the end points 'E' of all groups comprising the newly defined maximal circle happens to be less than or equal to the point (E) of the group which passes through the centre of the maximal circle. When one of the above conditions holds, first advance to the group in the next row which satisfies rule (1) then invoke step 1, (Fig. 3).

```
      P s s s s s s s s - - - - - - - - - - - - - P c c -
    - s s s s s s s s s - - -   - - - - - - - c c c c c -
  - - s s s s s s s s s - -       - - - - - c c c c c c c -
- - - s s s s s s s s s -           - - - c c c c c c c c c -
- - - - s s s s 0 s s s s             - - c c c c 0 c c c c - -
- - - s s s s s s s s s -           - - - c c c c c c c c c -
  - - s s s s s s s s s - -       - - - - - c c c c c c c -
    - s s s s s s s s s - - -   - - - - - - - c c c c c -
      s s s s s s s s s - - - - - - - - - - - - - c c c -
          Square                                 Circle
```

Fig. 2

A square/circle of size-1 first was defined on the pixel marked with 'P' and then enlarged to become maximal square/circle of size-9 with centre at '0'.

```
Columns                   1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2
      1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

R             - - - - - - - - - - - - - - - - P c c .
            P s s s s s s     - - -     - - c c c c c .
o         - s s s s s s s       -       - c c c c c c c
        - - s s s s s s s             - - c c c 0 c c c
w     - - - s s s 0 s s s - -       - - - c c c c c c c
        - - s s s s s s s - - -   - - - -   c c c c c -
s         - s s s s s s s - - - - - - -       c c c - - -
            s s s s s s s - - - - - -           - P s s s s
            - - - - - - . P c c - -               s s s s s -
              - - - - - c c c c c - -           - s s 0 s s - -
                - - -   c c 0 c c - - -       - - s s s s s - -
                  -     c c c c c - - - -   - - - s s s s s -
                        . c c c - - - - - - - - - - - - - - - -
```

Fig. 3

Circle Size	White pixels	Pixels in 1st group	C value for each group in a circle
1	0	1	0
2	0	2	0
3	1	3	0 0
4	1	2	1 1 0
5	3	3	1 1 1 0
6	4	2	1 2 2 1 0
7	6	3	1 2 2 2 1 0
8	5	4	1 2 2 2 2 1 0
9	7	3	2 2 3 3 3 2 2 0
10	8	4	1 2 3 3 3 3 2 1 0
11	8	5	1 2 3 3 3 3 3 2 1 0
12	9	6	1 2 3 3 3 3 3 3 2 1 0
13	11	7	1 2 2 3 3 3 3 3 2 2 1 0
14	12	6	1 2 3 4 4 4 4 4 4 3 2 1 0
15	13	5	2 3 4 4 5 5 5 5 5 4 4 3 2 0

-- Table 1 --

This Table shows 15 possible maximal circle sizes along with predetermined values. When enlarging a circle these values are used to determine whether or not a given group passes through a circle of size L+1. The 2nd column is used by MC2 algorithm in specifying the number of white pixels allowed in a maximal circle. The 3nd column specifies the number of pixels in the 1st group of the circle, while the rest of the columns specify the constant 'C' used in rule (3).

```
   Size 15            Size 14          Size 13         Size 12

    CCCCC              CCCCCC          CCCCCCC          CCCCCC
  CCCCCCCCC           CCCCCCCC        CCCCCCCCC        CCCCCCCC
 CCCCCCCCCCC         CCCCCCCCCC      CCCCCCCCCCC      CCCCCCCCCC
CCCCCCCCCCCCC       CCCCCCCCCCCC     CCCCCCCCCCC     CCCCCCCCCCCC
CCCCCCCCCCCCC      CCCCCCCCCCCCCC   CCCCCCCCCCCCC    CCCCCCCCCCCC
CCCCCCCCCCCCCCC    CCCCCCCCCCCCCC   CCCCCCCCCCCCC    CCCCCCCCCCCC
CCCCCCCCCCCCCCC    CCCCCCCCCCCCCC   CCCCCCCCCCCCC    CCCCCCCCCCCC
CCCCCCCCCCCCCCC    CCCCCCCCCCCCCC   CCCCCCCCCCCCC    CCCCCCCCCCCC
CCCCCCCCCCCCCCC    CCCCCCCCCCCCCC   CCCCCCCCCCCCC    CCCCCCCCCCCC
CCCCCCCCCCCCCCC    CCCCCCCCCCCCCC    CCCCCCCCCCC      CCCCCCCCCC
 CCCCCCCCCCCCC      CCCCCCCCCCCC     CCCCCCCCCCC       CCCCCCCC
 CCCCCCCCCCCCC       CCCCCCCCCC       CCCCCCCCC         CCCCCC
  CCCCCCCCCCC         CCCCCCCC         CCCCCCC
   CCCCCCCCC           CCCCCC
     CCCCC
```

Fig. 4

This Fig. shows how maximal circles look like according to Table 1.

Referring to Fig. 3, the letters 's', 'c' represent a pixel within a square/circle (e.g. 5 & 7 groups in Fig. 3); and the letter 'P' represents the size-1 square/circle which was initially defined on that pixel. After enlargement, the initial size-1 square/circle becomes a maximal square/circle of size 5 and 7.

The maximal squares (size 5 and 7 in Fig. 3) cannot be enlarged any more. That is so because of rule (2); where the first group of the second row in the pattern does not contain enough pixels to accommodate the larger maximal square of size-8, even though the first group in the ninth row has an enough number of pixels. The same condition applies to the size-5 maximal square on the bottom-right of Fig. 3; and in both cases no squares were defined by the remaining pixels (immediately after the pixel marked with 'P') in that group.

Since the 'E' point of the group passes through the centre of the maximal circle and the (rest of the) 'E' points of all other groups comprising the circle end before or at the same column, no advancement on the first group was made. Therefore those pixels marked with '.' in the 24rd and 25th columns were not included in this circle of Fig. 3. The reason for this is that if a circle were to be initialized and enlarged on the 23rd pixel, the largest possible circle to include those pixels is a maximal circle of size 2. If allowed, this could lead to a noisy centre-line.

Again, let us refer to the case of the circle with 5 groups at the bottom left of Fig. 3. A maximal circle of size 3 was defined on the first pixel of the first group in the ninth row. When a larger circle of size 5 was created on the second pixel, the size-5 circle happens to encompass the size-3 circle which already exists. In this case the size-5 circle replaces size-3 circle and those pixels marked with '.' were neglected, as were the case in the size-7 maximal circle on the top right.

All the four algorithms MS1, MS2, MC1, and MC2 employ the same syntax; with the exception that, MS2 and MC2 have slightly modified rules (2) and (3) respectively to allow a number of white pixels on the periphery of the maximal square/circle. In the MS2 algorithm, the number of permitted white pixels is determined by the length (of one side) of the maximal square (white pixels <=L/2); whereas in the case of MC2 algorithm, the number of white pixels is predetermined in advance in Table 1. Note that white pixels are only permitted on the perimeters of the square or circle (Fig. 5) in order to accommodate ruggedness along the edges. The centre-lines of a pattern produced by MS1, MS2, MC1, and MC2 algorithms are shown in Fig. 6.

A more detailed example on the following pattern in Fig. 7 & 8 may shed more light on the way the MC algorithms proceed from one group to the next. Whenever the parameter 'Y' changes in value, that means the algorithms jump from one row to another below it. Also look for a larger difference between 'X' values in the same row 'Y' which signifies the propagation from one group to another.

```
s s s s s s B            c c c
s s s s s s s          c c c c B
B s s s s s s        c c c c c c B
B s s 0 s s s        c c c 0 c c c
s s s s s s s        B c c c c c c
s s s s s s B          c c c c B
s s s s s s B            c c B
```

Fig. 5

Maximal square/circle with few white pixels(B).

```
          - - -                     - - -
- -     - - 0 - 0 - -     - -     - - 0 - - - -
 0       0         0                         0
- - - - - - - -   - -     - 0 - - - 0 0 0   - -
 0     0     0
- - 0 - -   - - -         - 0 0 0 -   - - -
               0                         0
  - - -       - -           - - -       - -
__________ MS1 ____________________ MS2 _______

          - - -                     - - -
- -     - - 0 - 0 - -     - -     - - - - - - -
 0       0         0                 0 0     0
- - - - - - - -   - -     - 0 - - 0 - - -   - -
 0     0     0               0
- - 0 - -   - - -         - - - 0 -   - - -
               0                         0
  - - -       - -           - - -       - -
__________ MC1 ____________________ MC2 _______
```

- Fig. 6 -

Sample outputs of MS1, MS2, MC1, and MC2.

```
A CENTER AT Y, X   4 36  OF SIZE  4  GROUPS
A CENTER AT Y, X   4 38  OF SIZE  4  GROUPS
A CENTER AT Y, X   4 40  OF SIZE  4  GROUPS
A CENTER AT Y, X   4 42  OF SIZE  4  GROUPS
A CENTER AT Y, X   5 33  OF SIZE  3  GROUPS
A CENTER AT Y, X   5 45  OF SIZE  3  GROUPS
A CENTER AT Y, X   6 48  OF SIZE  4  GROUPS
A CENTER AT Y, X   7 51  OF SIZE  5  GROUPS
A CENTER AT Y, X   7 29  OF SIZE  3  GROUPS
A CENTER AT Y, X  10 54  OF SIZE  6  GROUPS
A CENTER AT Y, X   9 25  OF SIZE  3  GROUPS
A CENTER AT Y, X  13 57  OF SIZE  7  GROUPS
A CENTER AT Y, X  16 58  OF SIZE  8  GROUPS
A CENTER AT Y, X  14 22  OF SIZE  4  GROUPS
A CENTER AT Y, X  18 58  OF SIZE  8  GROUPS
A CENTER AT Y, X  18 20  OF SIZE  6  GROUPS
A CENTER AT Y, X  20 58  OF SIZE  8  GROUPS
A CENTER AT Y, X  21 21  OF SIZE  7  GROUPS
A CENTER AT Y, X  21 61  OF SIZE  7  GROUPS
A CENTER AT Y, X  24 20  OF SIZE  8  GROUPS
A CENTER AT Y, X  23 23  OF SIZE  7  GROUPS
A CENTER AT Y, X  28 18  OF SIZE  8  GROUPS
A CENTER AT Y, X  24 28  OF SIZE  4  GROUPS
A CENTER AT Y, X  24 52  OF SIZE  4  GROUPS
A CENTER AT Y, X  30 18  OF SIZE  8  GROUPS
A CENTER AT Y, X  25 31  OF SIZE  3  GROUPS
A CENTER AT Y, X  25 33  OF SIZE  3  GROUPS
A CENTER AT Y, X  25 35  OF SIZE  3  GROUPS
A CENTER AT Y, X  25 37  OF SIZE  3  GROUPS
A CENTER AT Y, X  25 39  OF SIZE  3  GROUPS
A CENTER AT Y, X  25 41  OF SIZE  3  GROUPS
A CENTER AT Y, X  25 43  OF SIZE  3  GROUPS
A CENTER AT Y, X  25 45  OF SIZE  3  GROUPS
A CENTER AT Y, X  25 47  OF SIZE  3  GROUPS
A CENTER AT Y, X  25 49  OF SIZE  3  GROUPS
```

Fig 7

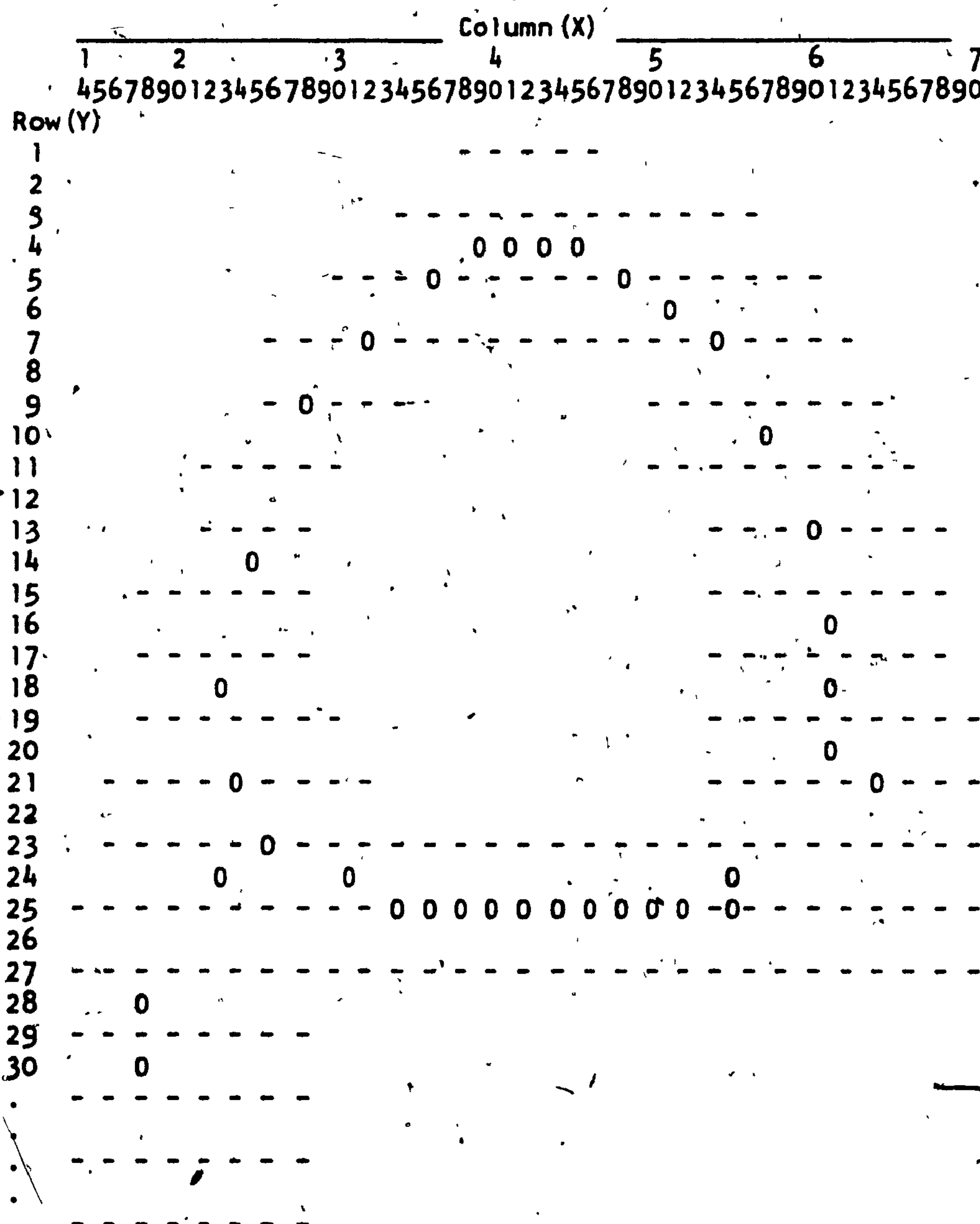


Fig. 8

'O' represents the centre of a maximal circle.

4 Reconstruction of the original pattern

A database of patterns may be stored on disk or tape by saving the link-list which reresents the B and E points of all groups comprising the patterns instesd of storing the patterns in the form of matrices. This method saves storage space because binary patterns are usually composed of large sparse matrices. The reconstruction of the original pattern [1], [11], [12], [13] is easily achieved by the manipulation of the B and E points. When the need to recreate the original pattern arises, a temporary matrix can be formed and the interval between the points B and E can be filled for all groups in each row of the matrix, producing an exact copy of the original pattern.

5 Filling algorithm

In the proposed MS and MC algorithms for finding the centre-lines of binary patterns, it should be noted that, a maximal square/circle centre may not always lie on a pixel and the centres of two overlapping squares/circles may not be adjacent to each other (discontinuity). This is due to two factors :

(1)--: Given the hardware, the centre of a maximal square/circle of size 2 (4 pixels) for example, being equidistant from the 4 points, cannot be addressed in digital co-ordinates. Hence, in order to obtain the exact centre-lines of a pattern, we expanded the pattern so that the centre of any maximal square/circle will fall exactly on the centre of all the pixels it encompasses. See illustrations in Fig. 9.

(2)--; Overlapping maximal squares/circles are not always equal in size (due to the shape of the given pattern).

In order to maintain a connected centre-line, and overcome the above two obstacles, we developed a filling algorithm which uses a 66-pixel window. The size for this window is so chosen that it covers the longest path between two centres of the maximal squares/circles encountered.

```
Original pattern              Expanded pattern
   not filled                     not filled

                                     - - -

                          - -     - - 0 - 0 - -
      ---                 0       0          0
 0-  0-0-00-              - - - - - - - - -   - -
 0--0--0- --              0     0     0
 --0-- -0-                - - 0 - -     - - -
  ---   --                               0
                             - - -        - -
```

```
Original pattern              Expanded pattern
     filled                         filled

                                     - - -

                          - -     - 0000000 - -
      ---                 0       0  0     000
 0-  000000-              -0- - - 0 - -0-    - -
 0--0--0- --              0       0     0
 -00-- -0-                - 00000 -   - 0 -
  ---   --                              0
                             - - -        - -
```

Fig. 9

```
 _______________________________
| | | | | |C|*|*|*|*|*|
|-+-+-+-+-+-+-+-+-+-+-|
| | | | | | | | | | | |
|-+-+-+-+-+-+-+-+-+-+-|
| | | | | | | | | | | |
|-+-+-+-+-+-+-+-+-+-+-|
| | | | | | | | | | | |
|-+-+-+-+-+-+-+-+-+-+-|
| | | | | | | | | | | |
|-+-+-+-+-+-+-+-+-+-+-|
| | | | | | | | | | | |
|-+-+-+-+-+-+-+-+-+-+-|
```

Fig. 10

66-Pixel window

5.1 *Filling strategy*

The following steps were performed using the 66-pixle window (Fig. 10) to fill the gaps of centre-lines (in both cases original and expanded patterns).

<u>Step-1</u>

The window moves from one maximal square/circle to the next in the direction of left to right and top to bottom covering all points in the centre-lines, and aligning the current maximal square/circle centre to the point 'C' in the window.

<u>Step-2</u>

If a maximal square/circle centre occurs in the window point marked with an '*', then fill the gap between 'C' and '*'.

<u>Step-3</u>

Excluding the first row of the window, search for a centre which has the shortest path to the point 'C'. If this is found, fill the gap. Otherwise do nothing (this point of the centre-line is an end-point). If two centres having equal distance (they are in the same row) from the point 'C' in the window and the distance between these two centres is greater than 3 pixels, fill both paths (the centre-line is diverging); otherwise, fill only the gap between the point 'C' and the left centre in the window.

<u>Step-4</u>

Repeat steps 1-3 until all centres have been covered.

An example of the filling algorithm is shown on the following pages for both the original and expanded patterns of the digitized letter 'A' (Figs. 11-14). It should be noted that, the original pattern uses a smaller window-size of 28-pixels.

```
                    ---
                  -------
                ----------
               ---0-000-----
              -----0----0----
             ----0------00---
             ---0-------0---
            --00----   ---0---
            --0----     --0----
            --0---      ---0---
           --0----       --0---
           --0---        -------
           --0---        ---0---
           ------        ---0---
           --0--         ---0----
          --0---         ---0----
          --0---         ----0-----
          --0---   -----0-0-0----
          ------------------------
          ---0-0000-0-00-------0---
          --0---------------------
         --0-------------    ---0---
         ------    ---       ---0---
         --0--               ---0----
         --0--               ----0---
        ---0---               ---0---
        --0----               ---0----
        --0---                ---00---
        --0---                ----0---
       -------                 ---0---
      ----0---                  --0---
     ----0----                  -------
     ---0--0---                 ---0---
    0---0-------0---        0----0----0--
    --00----------0-        --000--------0000-
     --------------          ---------------
```

The centre-lines are displayed on top of the original pattern. No filling is done.

Fig. 11

```
                 ---
               -------
             ----------
            ---00000-----
           ----0----0----
          ----0------00---
          ---0--------0---
         --00----    ---0---
         --0----      --0----
         --0---       ---0---
        --0----        --0---
        --0---         --0----
        --0---         ---0---
        --0---         ---0---
        --0--          ---0----
       --0---          ---0---
       --0---         -----0----
       --0---     -----000-0----
       --0-----------0-----0---
       ---00000000000-------0---
       --0-----------------0----
      --0------------    ---0---
      --0---     ---     ---0---
      --0--              ---0----
      --0--              ----0---
    ---0---              ---0---
    --0----               ---0----
    --0---                ---00---
    --0---               -----0---
   ---0---                 ---0---
  ----0---                 --0---
 ----00---                 --0----
 --00--0---                -0-0---
0---00-----00---        0----0---00--
-000---------00-        -0000-------00000-
----------------         ----------------
```

The gaps in the centre-lines are
filled to maintain continuity

Fig. 12

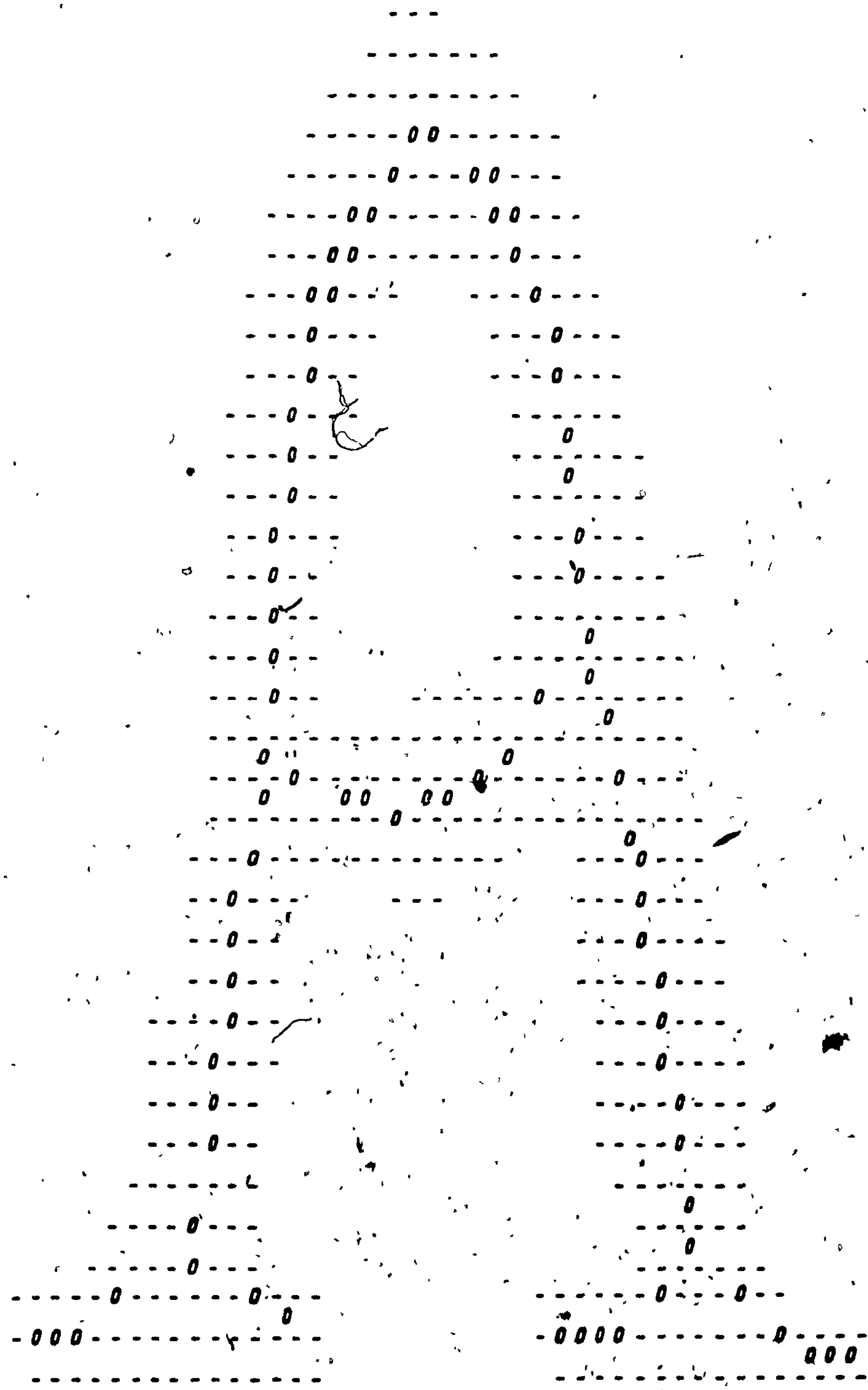


Fig. 13

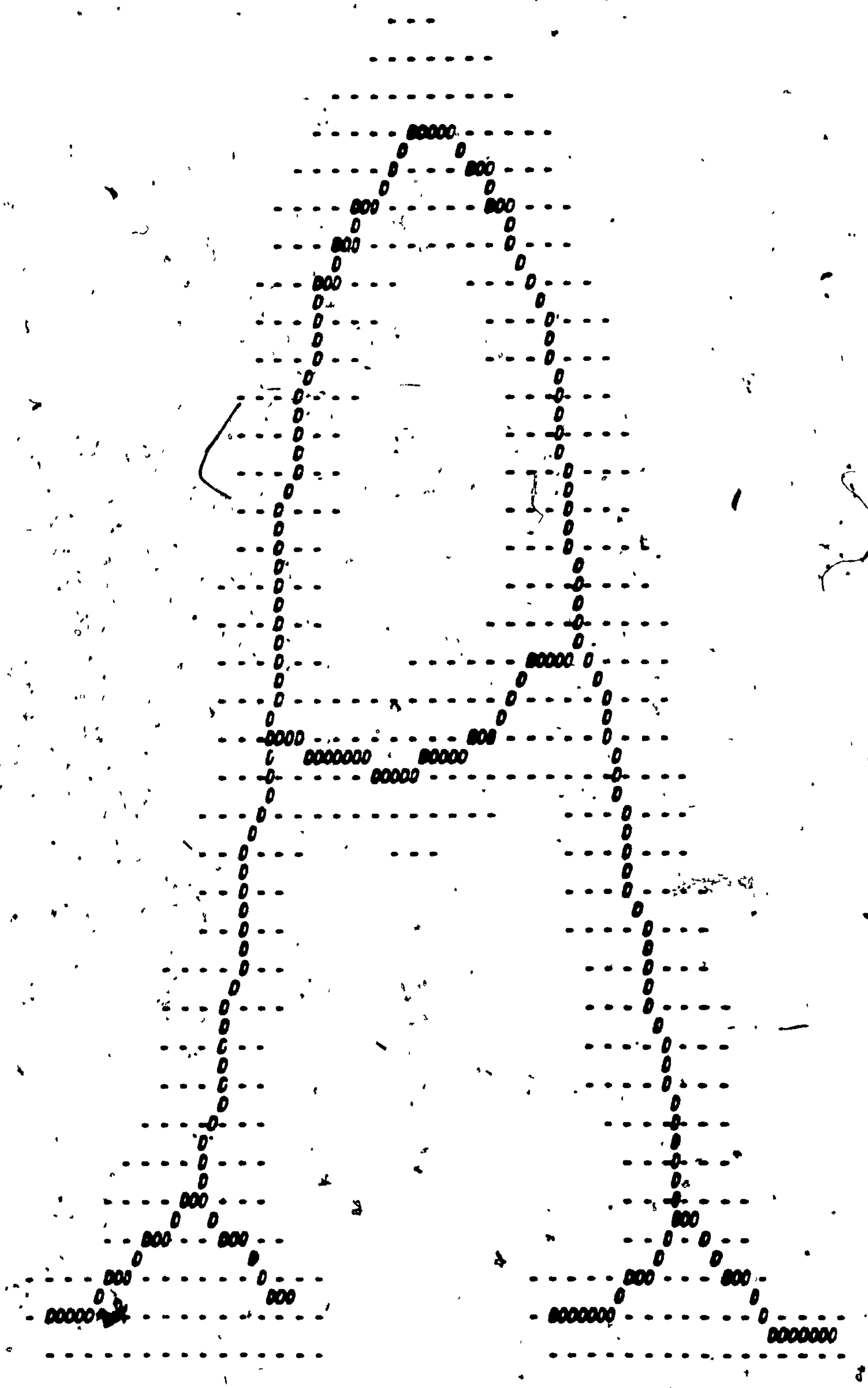


Fig. 14

The expanded pattern and its centre-lines

6 Implementation of the Algorithms and Experimental Results

The programs for MS1, MS2, MC1 and MC2 algorithms were coded in Pascal and executed on a CDC/Cyber 830D Computer. Each program processed the same set of binary patterns; and the results are presented in the Appendices. In those results, the centre-lines were overlaid on top of the original patterns so that we can see clearly how good the centre-lines are for the given pattern produced by the different algorithms and compare them. More importantly, we can also see how close each centre-line represents the shape of the pattern. As with most thinning algorithms, the problem of centre-line discontinuity is inherent in the algorithms; this was overcome by using smoothing and connecting procedures developed in this work.

The results produced by all four algorithms, in terms of centre-line smoothness, resistance to noise and resemblance to the original pattern, can be rated from fair to very good by visual inspection. We note that MS1 and MC1 (no white pixels allowed), even though they are vulnerable to noise, produce centre-lines of good quality with the least amount of distortion when the input pattern is composed of thick patches. On the other hand, MC2 algorithm generally produces smooth centre-lines. By comparing the results of MS1 and MS2, one can see that MS2 produces centre-lines with less noisy branches than MS1 (e.g., see Pages 30 and 42 32 and 44). But when the beginning and the ending points of the groups produce a zig-zag pattern, maximal squares which contain white pixels may yield noisy centre-lines such as double centre-lines, refer to pages 44, 45, 47, 50, 52.

Further analysis of the results reveal that MC1 and MC2 produce much smoother centre-lines than MS1 and MS2. However when patterns have straight vertical and horizontal edges, then by the nature of the square algorithms, MS1 and MS2 produce better results, see e. g. the centre-lines of the character H on pages 31, 43, 55, 67 and the centre-lines of the Chinese character shown on pages 35, 47, 59, 71.

Since our objective focusses on the development and comparison of these algorithms in the areas of centre-line smoothness, resistance to noise and resemblance to the original pattern, the emphasis on CPU time was not considered as were most algorithms developed by other research groups [7, 9, 14, 15, 16, 18, 19]. We may add that these algorithms are fairly efficient (see Table 2), given the fact that a fair amount of processing time was spent by the smoothing algorithm. But this can be improved if further attention were given to this area.

In terms of memory space; a dynamic storage allocation was employed by the algorithms to store the original pattern, plus a temporary matrix to display the centre-lines on top of the pattern, and few local variables used by the algorithms.

In general, in terms of smoothness, resistance to noise and resemblance to original pattern, we rank the algorithms in this order : MC2, MC1, MS1, and MS2; MC2 being the best.

Algorithm	Without smoothing	With Smoothing
	CPU Time	CPU time
MS1	35.245 Sec	41.074 Sec
MS2	33.303 Sec	38.873 Sec
MC1	46.857 Sec	53.681 Sec
MC2	61.646 Sec	67.735 Sec

Table 2.

CDC Cyber/830D Pascal, the CPU time refers to the total amount of time in seconds required by the computer to process all 18 patterns.

7 Concluding Remarks

Two types of algorithms were proposed and implemented for centre-line tracing of binary pictures. In both the maximal squares and maximal circles algorithms, some white pixels are allowed to account for shape variabilities of the digitized pattern.

In general, these algorithms tend to favor patterns which are composed of mainly relatively thin lines, not patterns composed of thick blocks. Under most circumstances however, the MC2 algorithm produces the best centre-lines of all the four algorithms presented in this work and most other algorithms presented by various research groups. This is due to two factors :

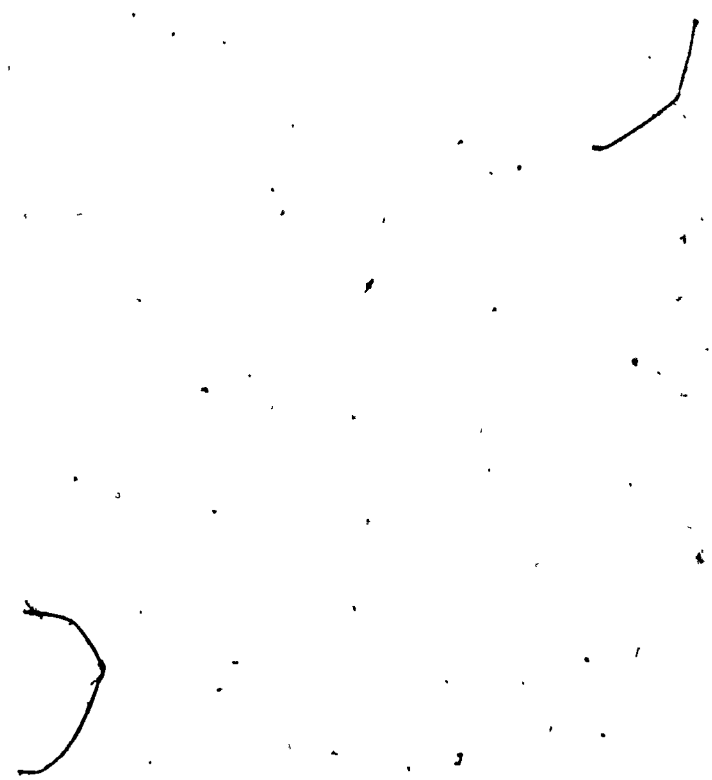
(1) The expansion of the original pattern aids in making the centre-lines fall exactly in the middle of the pattern.

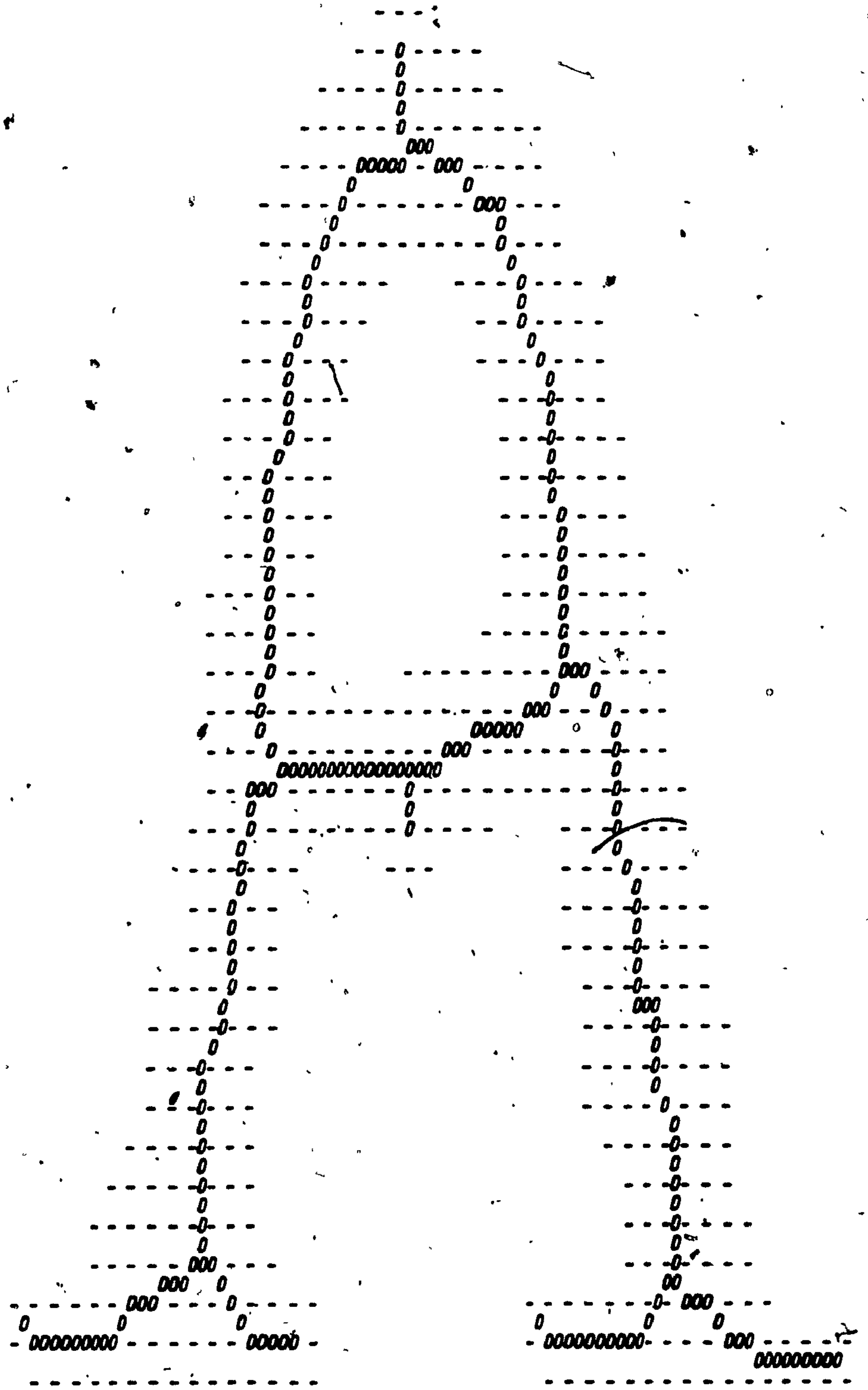
(2) The inclusion of a few white pixels on the periphery of the maximal circle helps in obtaining smoother centre-lines.

The reconstruction of the original pattern is easily achieved by the manipulation of the B and E points of the pattern, with no loss of detail. Finally, by examining the results of MS1, MS2, MC1 and MC2 algorithms, one can see that, in general MC1 and MC2 pruduce smoother centre-lines than MS1 and MS2 algorithms.

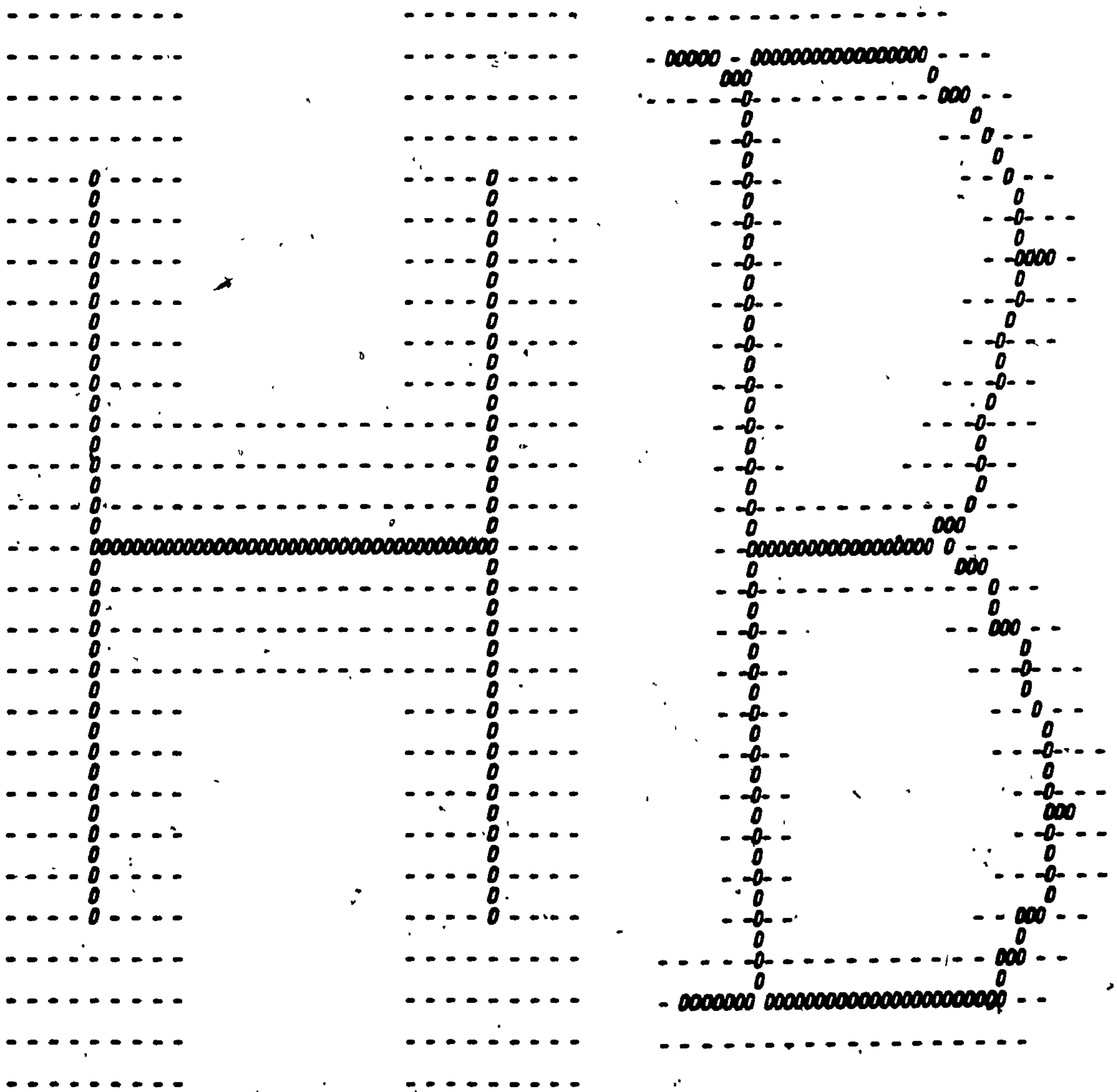
8 APPENDICES

Centre-Lines produced by the MS1 Algorithm

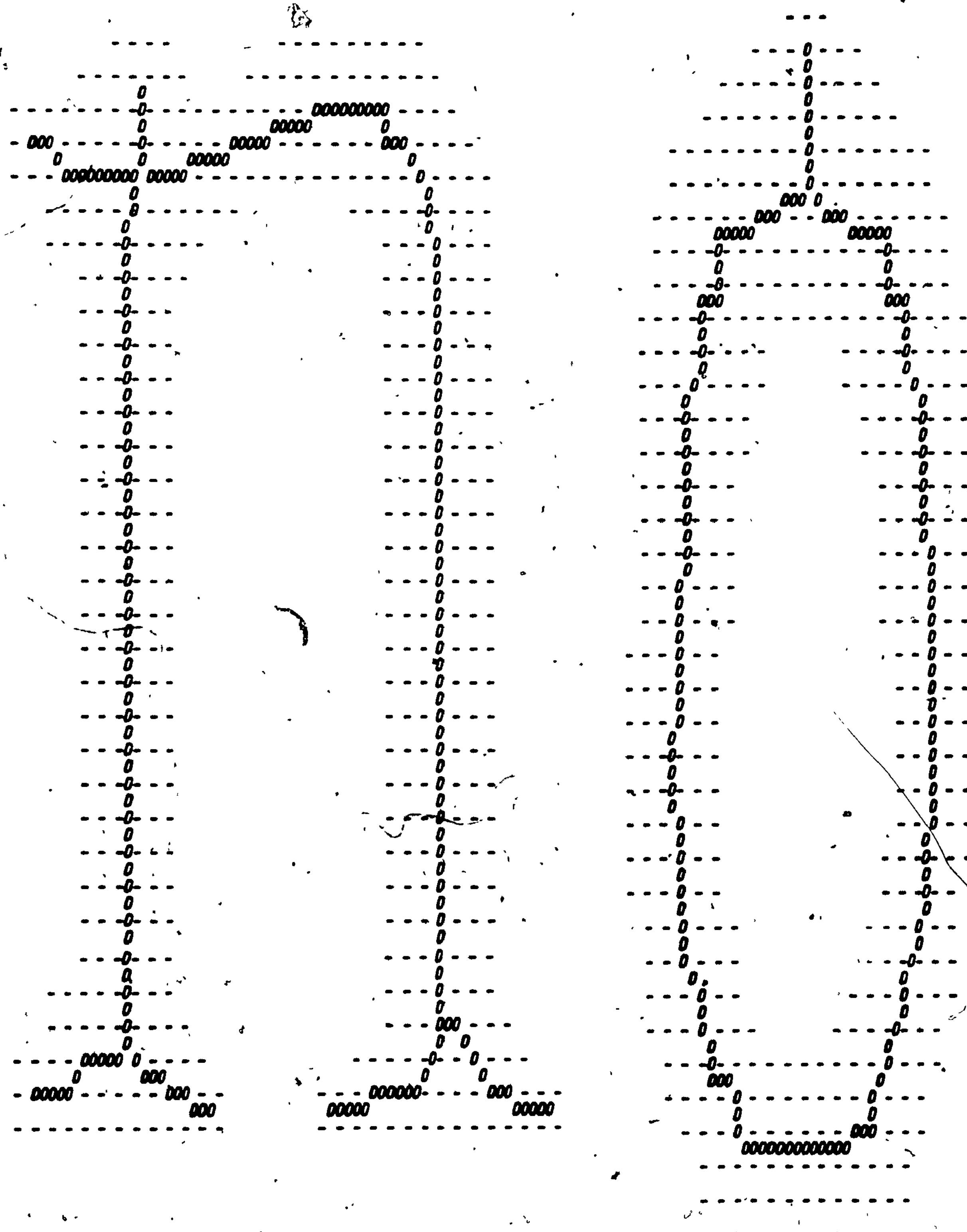




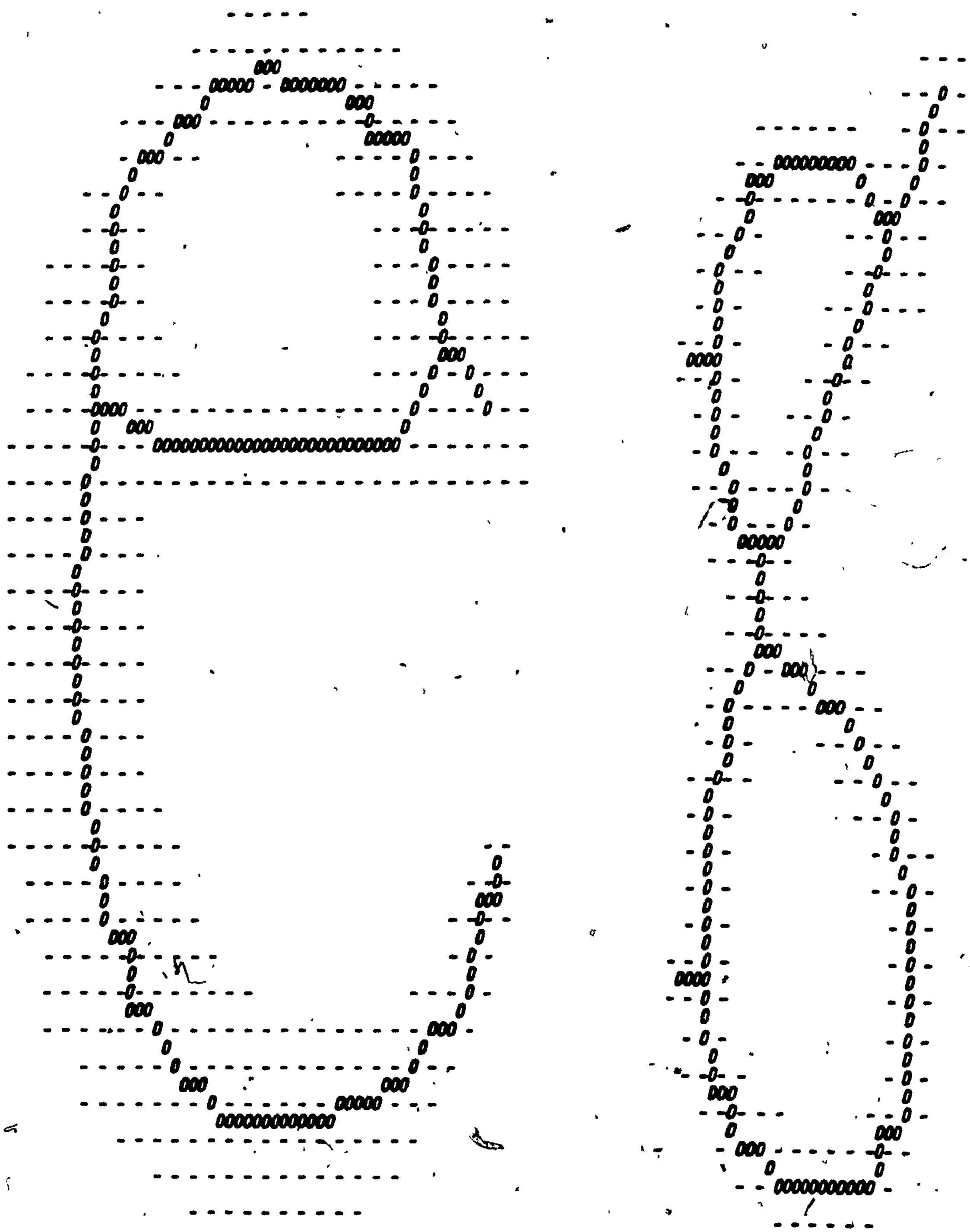
MS1 centre-lines



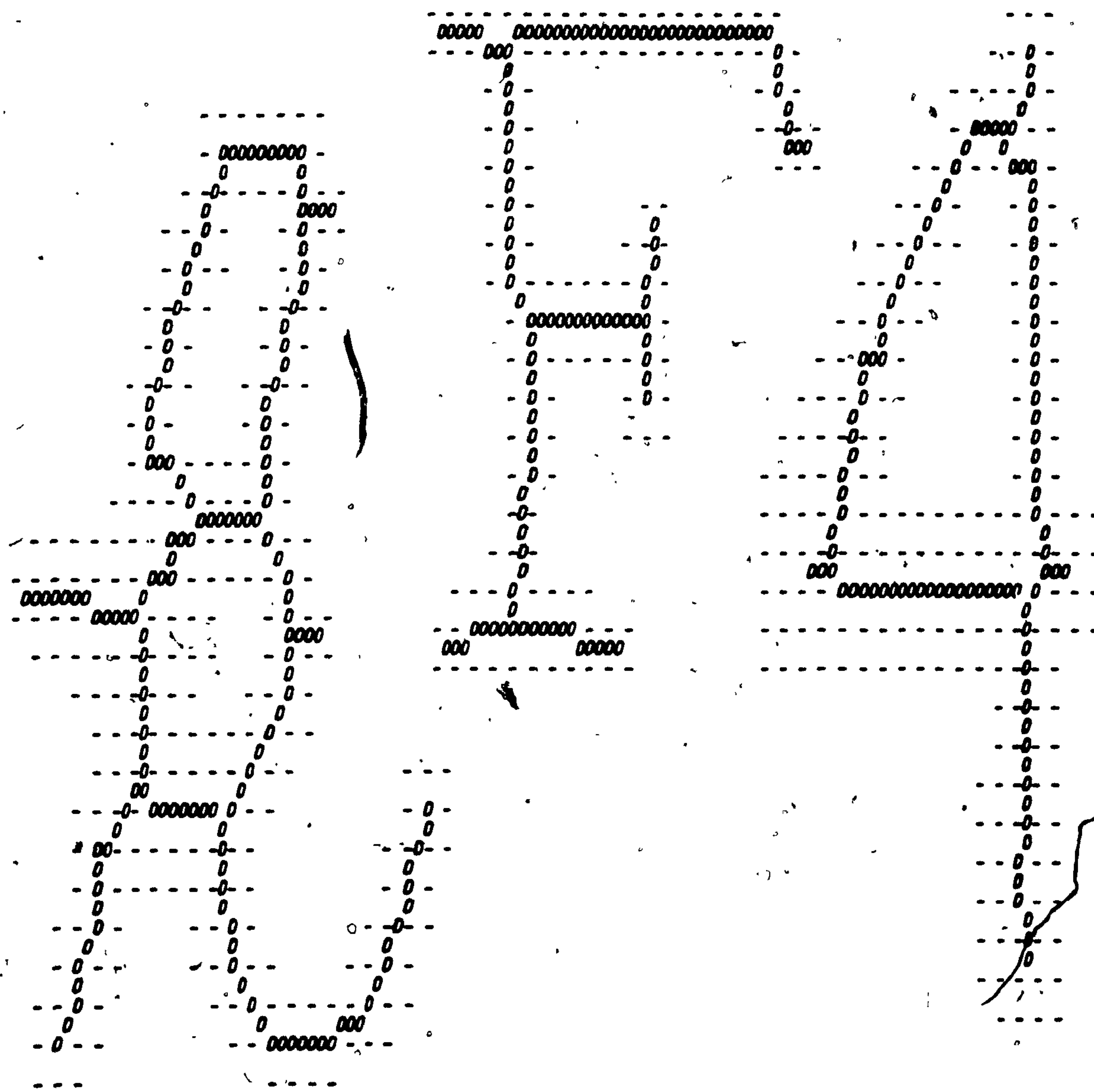
MS1 centre-lines



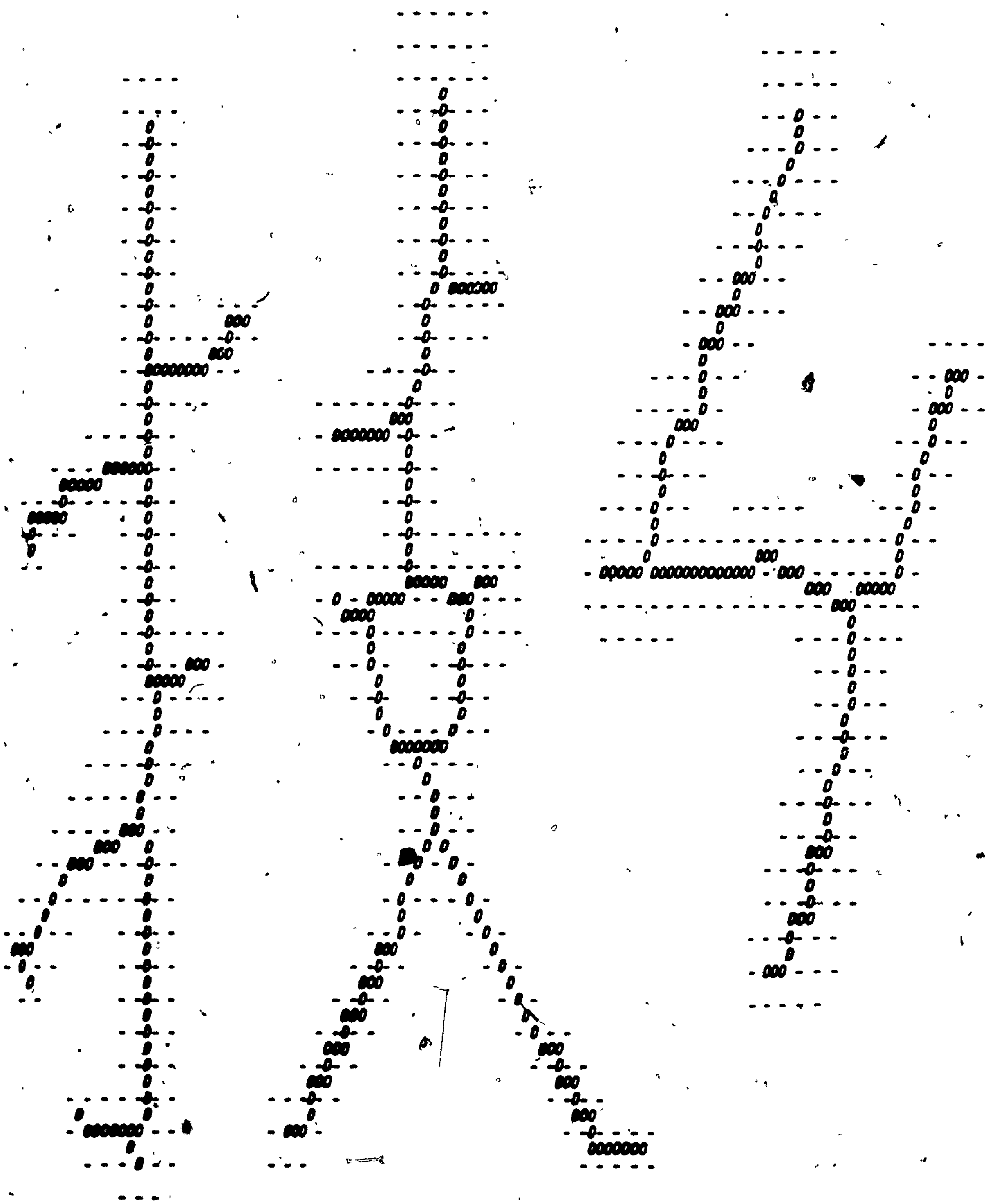
MS1 centre-lines



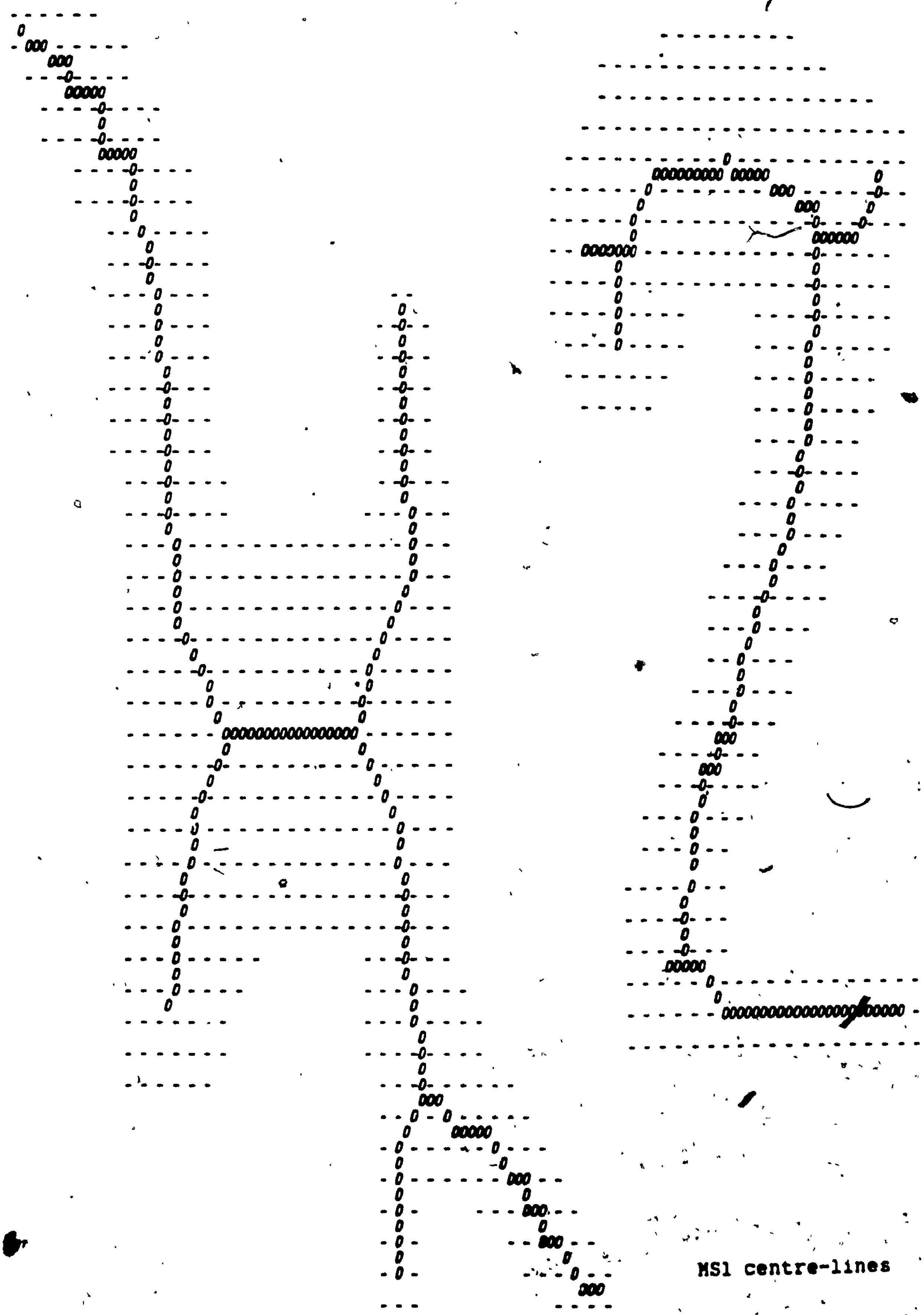
MS1 centre-lines



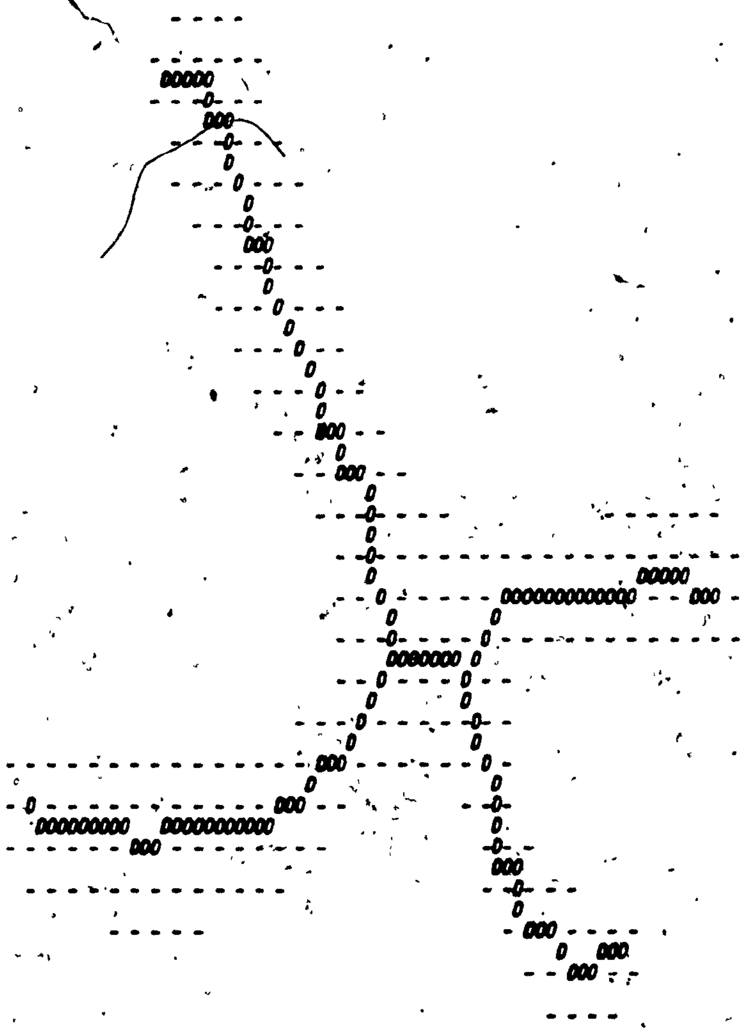
MS1 centre-lines



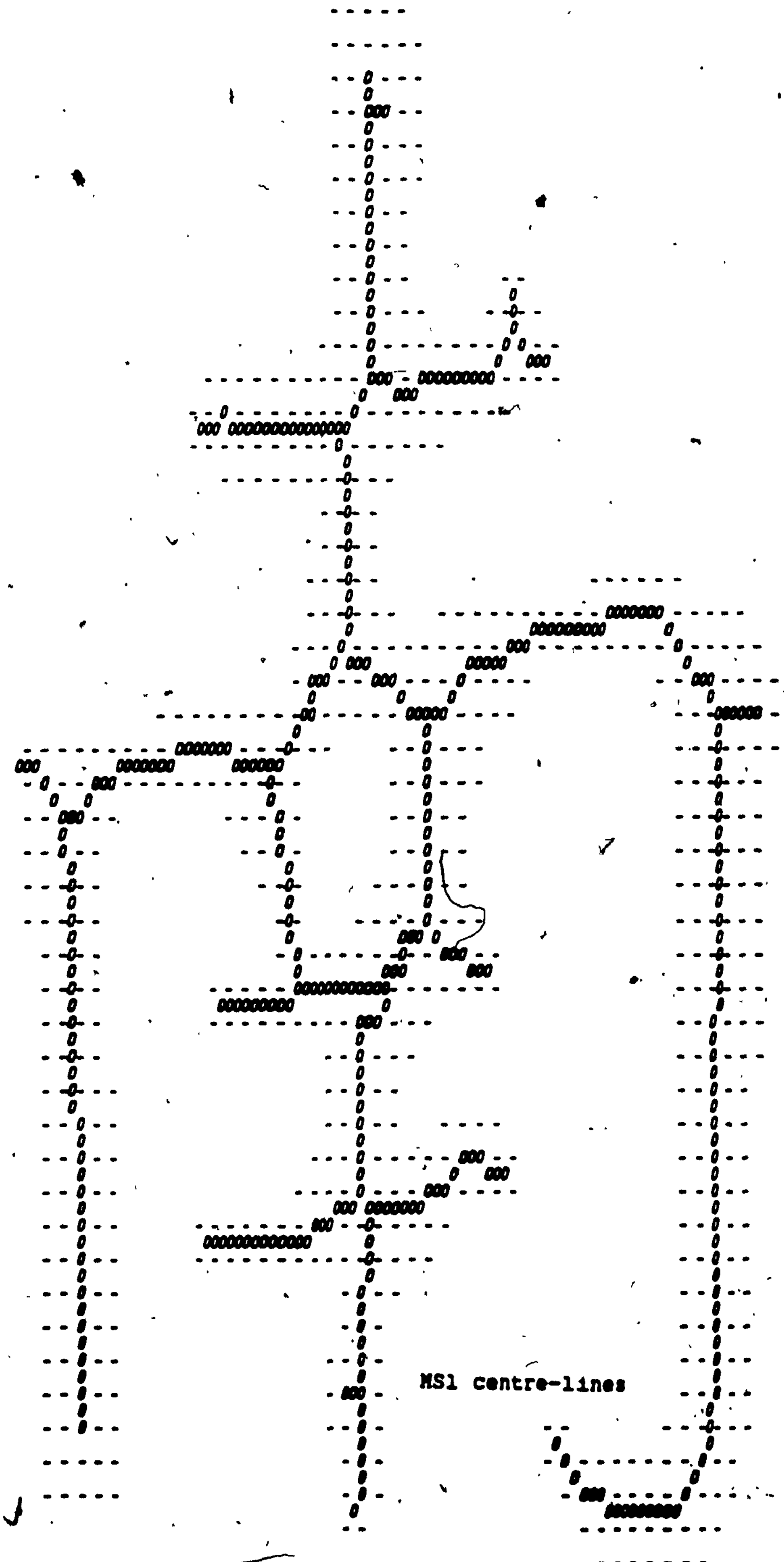
MS1 centre-[illegible]s



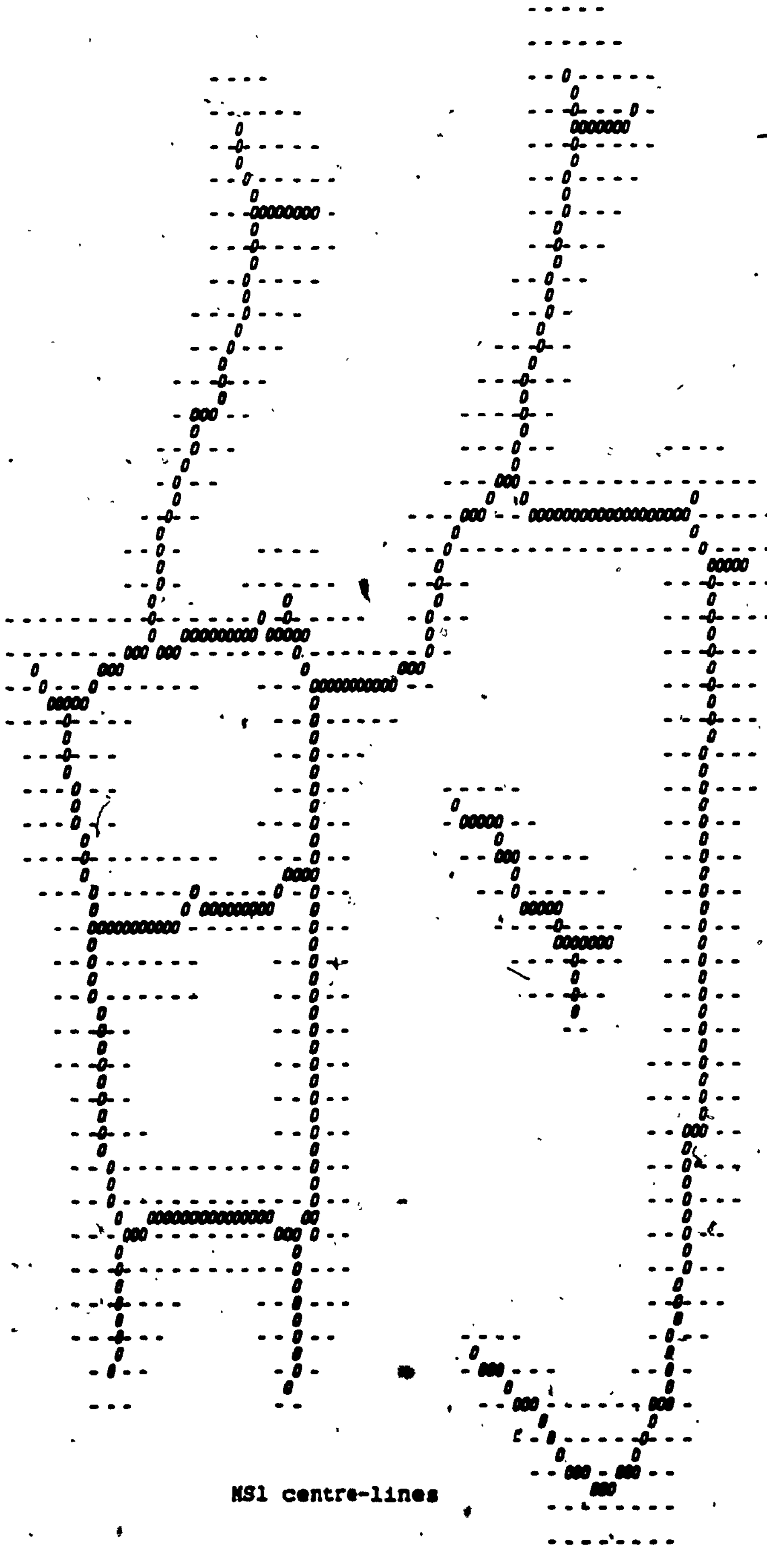
MS1 centre-lines



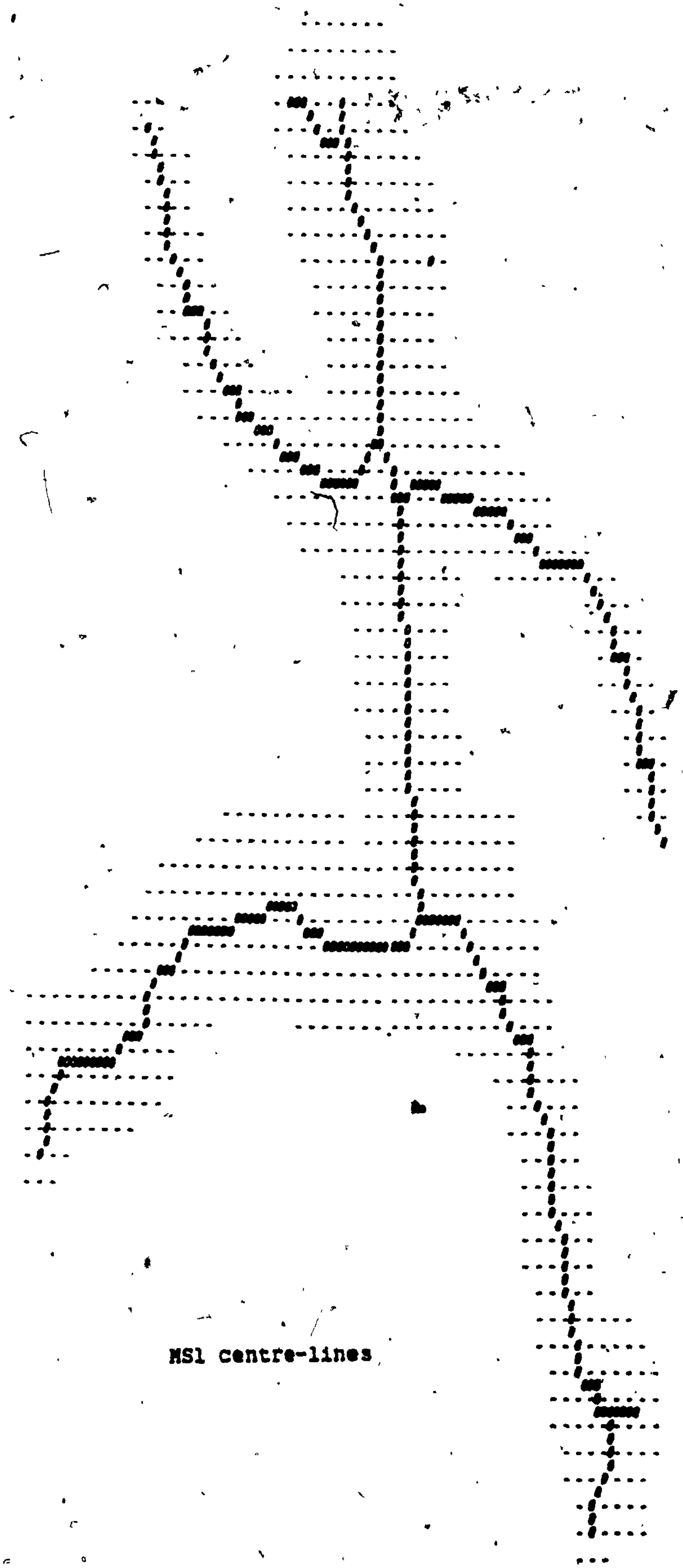
MS1 centre-lines



MS1 centre-lines

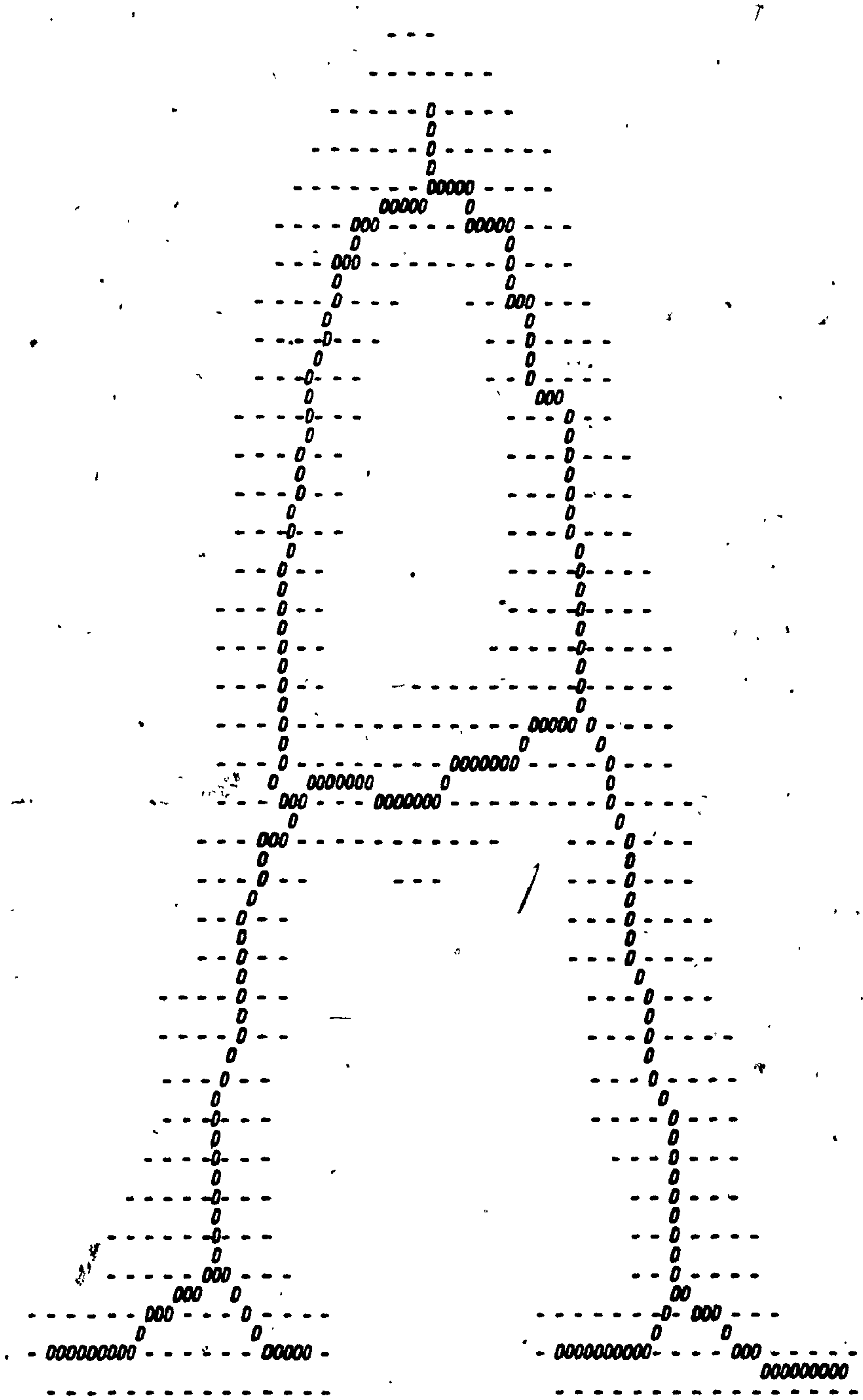


MS1 centre-lines

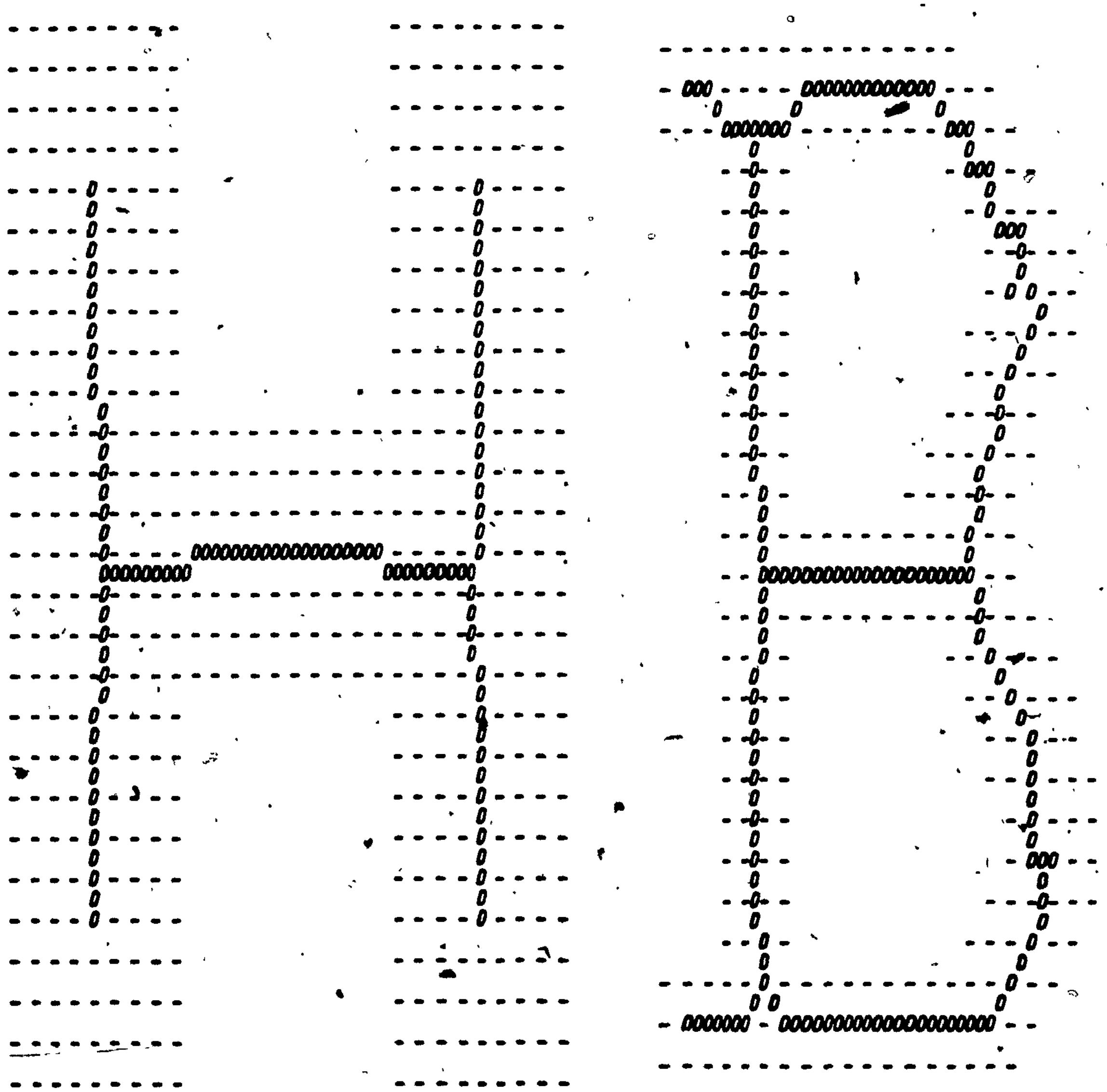


MS1 centre-lines

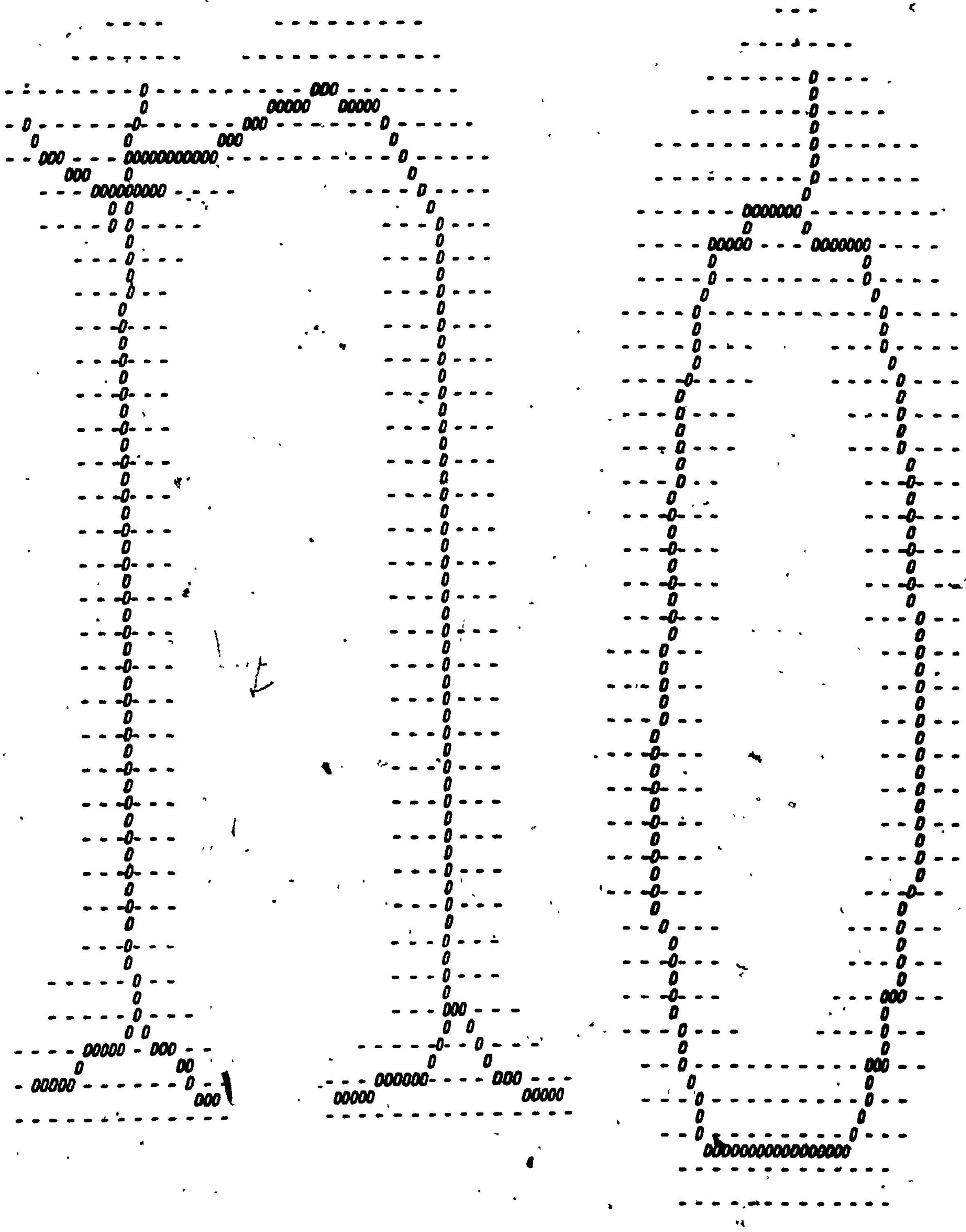
Centre-Lines produced by the MS2 Algorithm



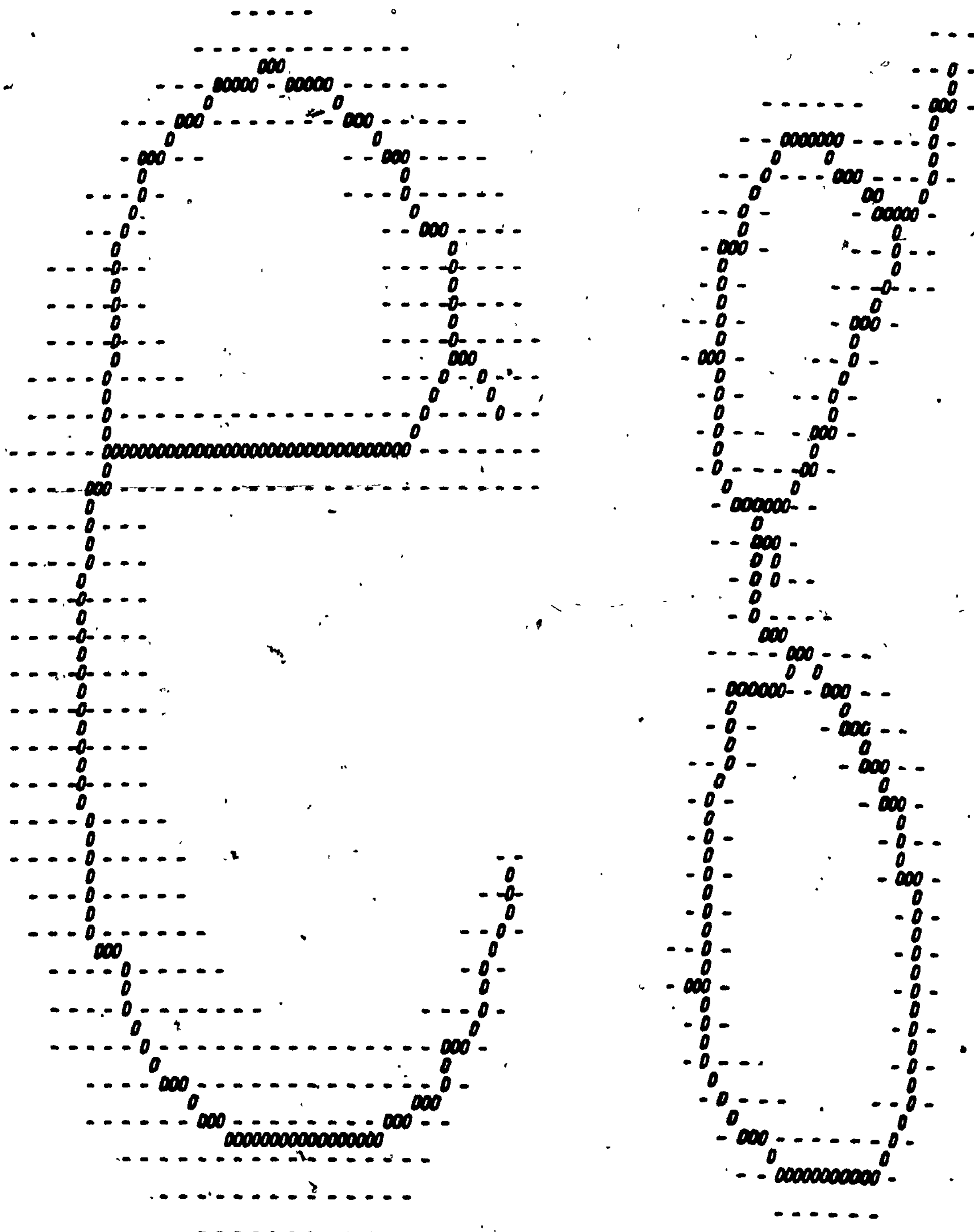
MS2 centre-lines



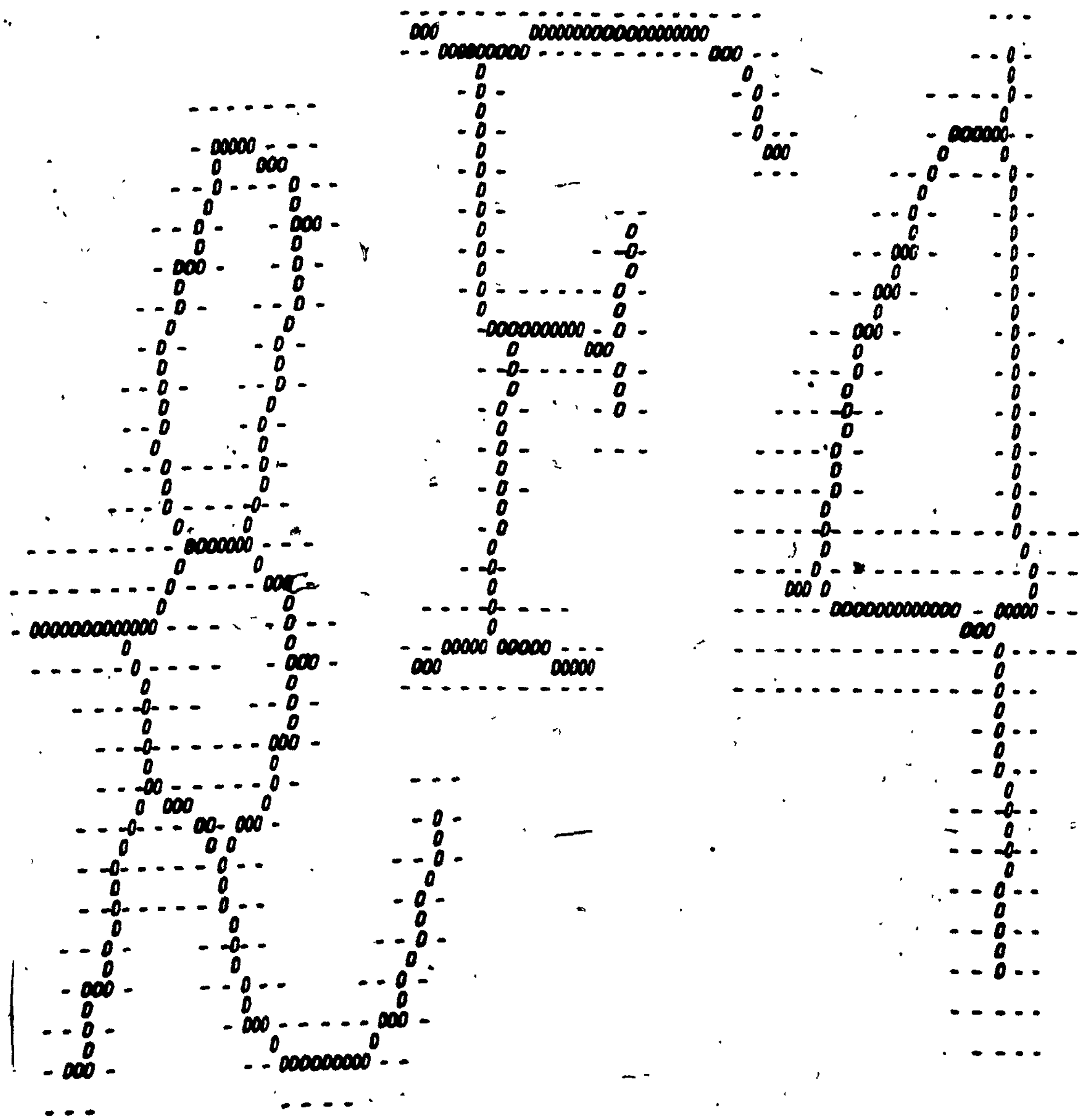
MS2 centre-lines



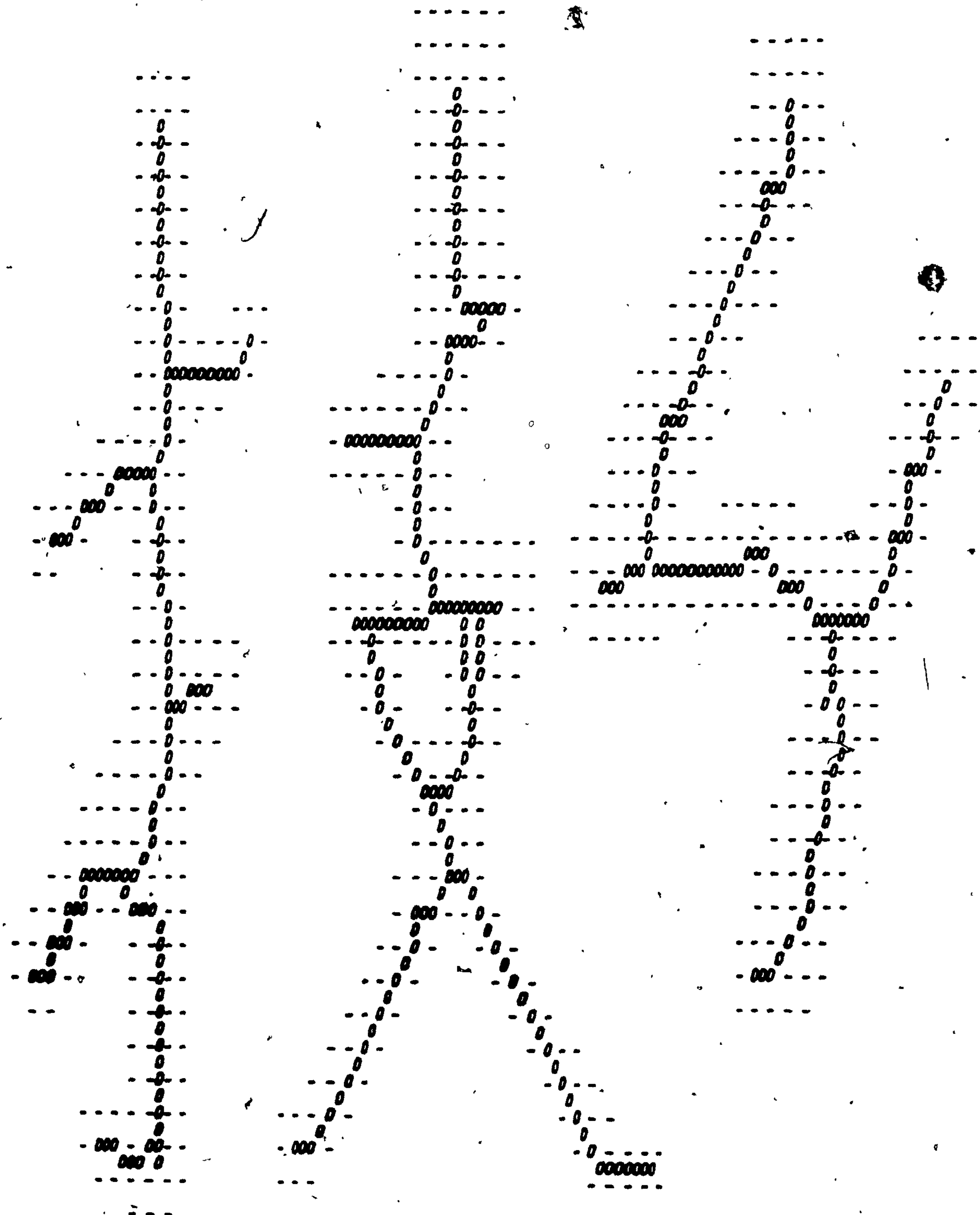
MS2 centre-lines



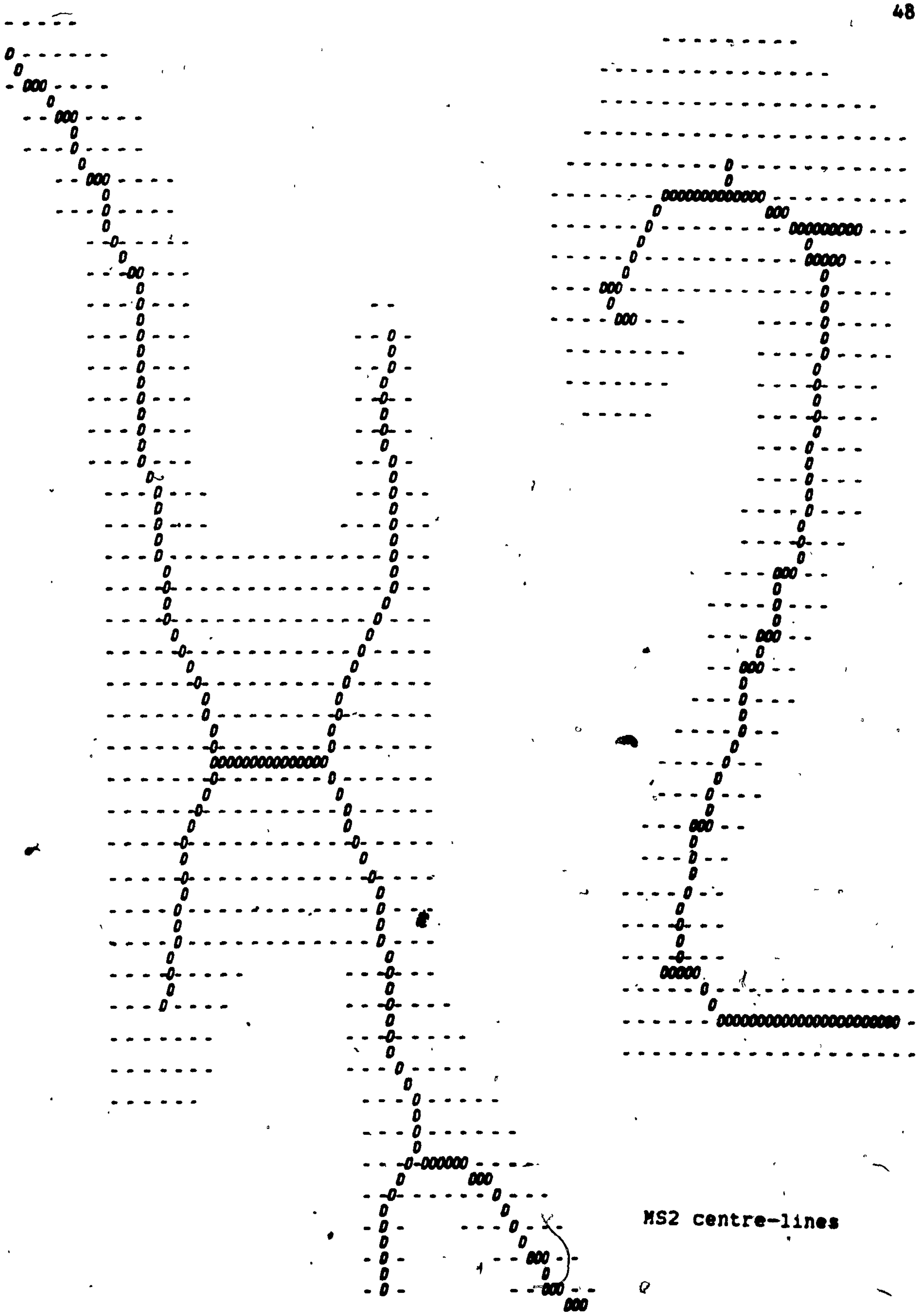
MS2 centre-lines



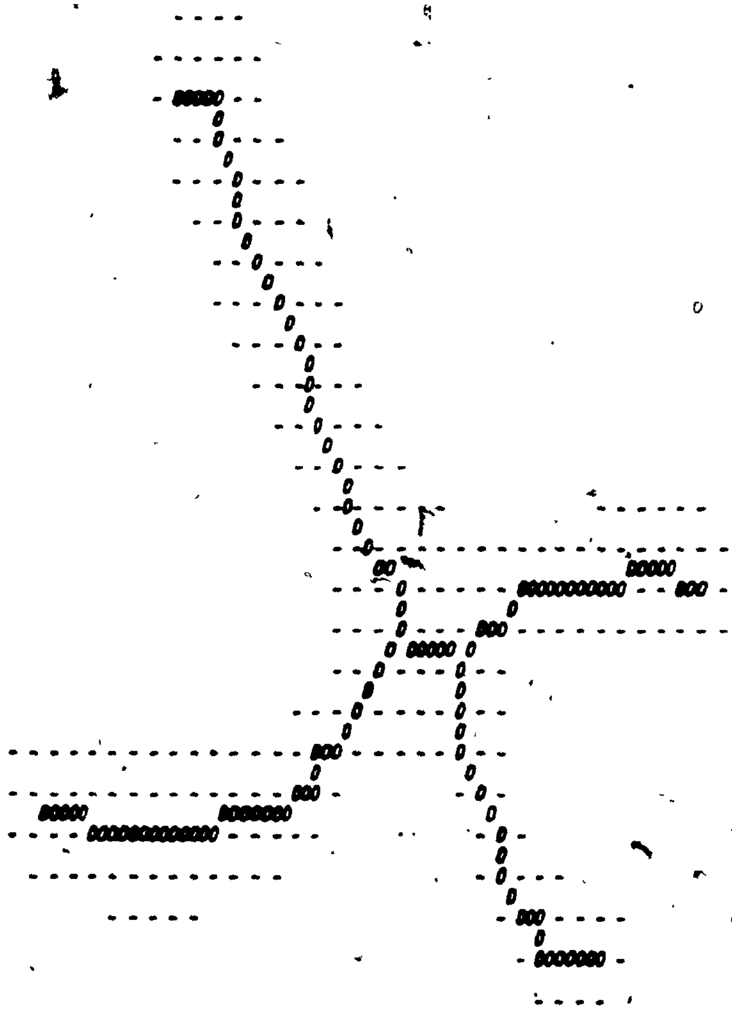
MS2 centre-lines



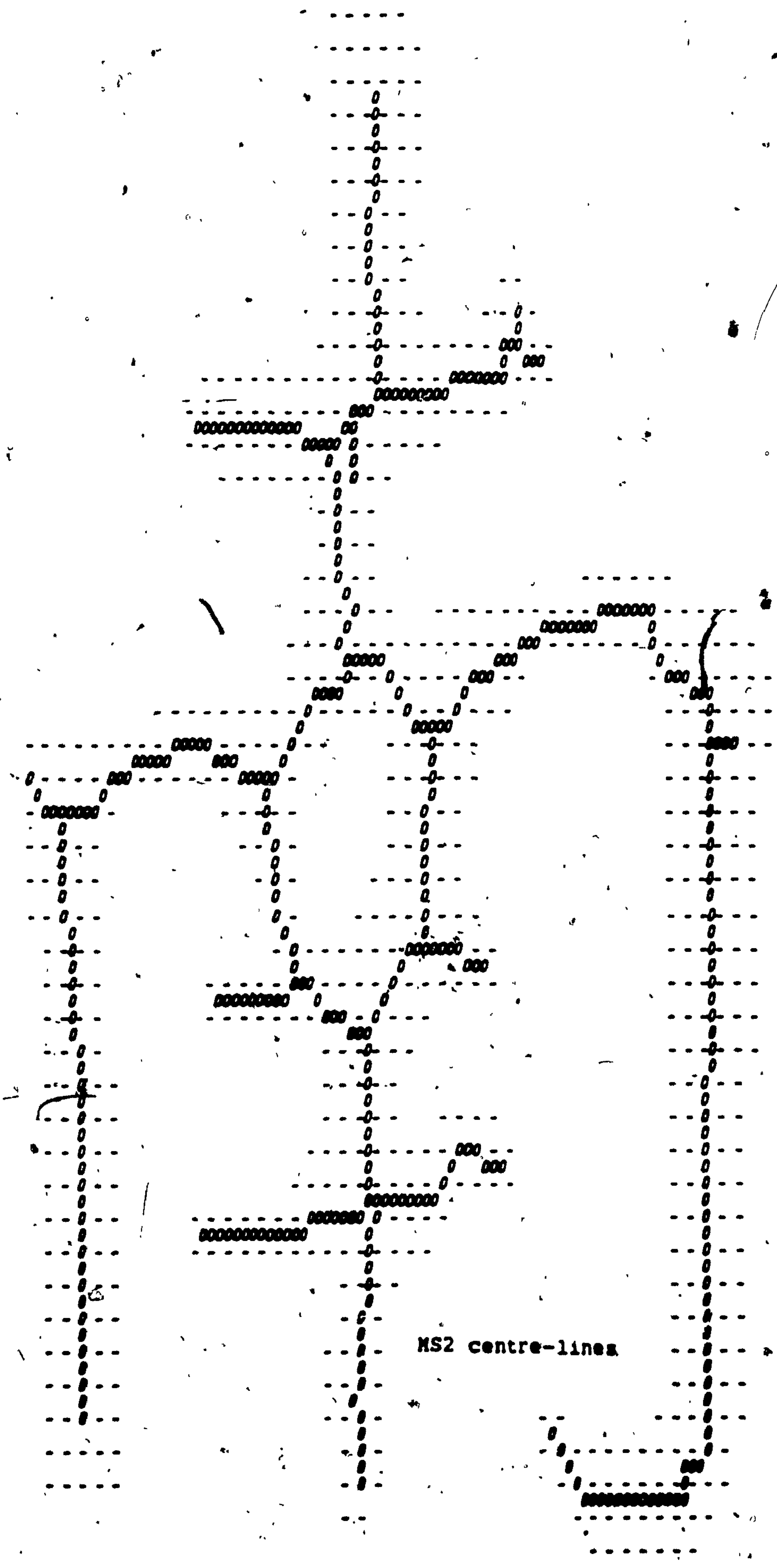
MS2 centre-lines



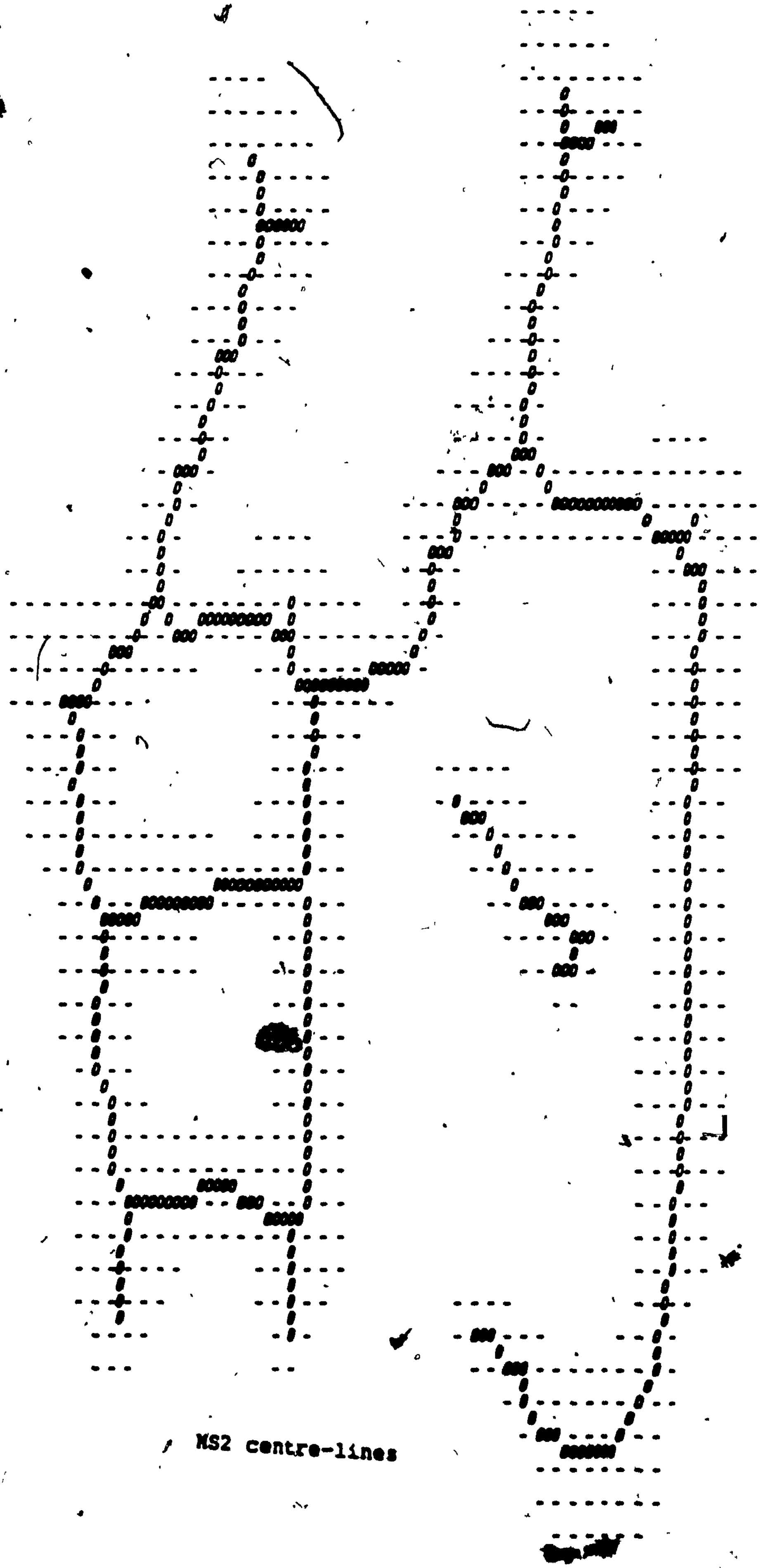
MS2 centre-lines



MS2 centre-lines



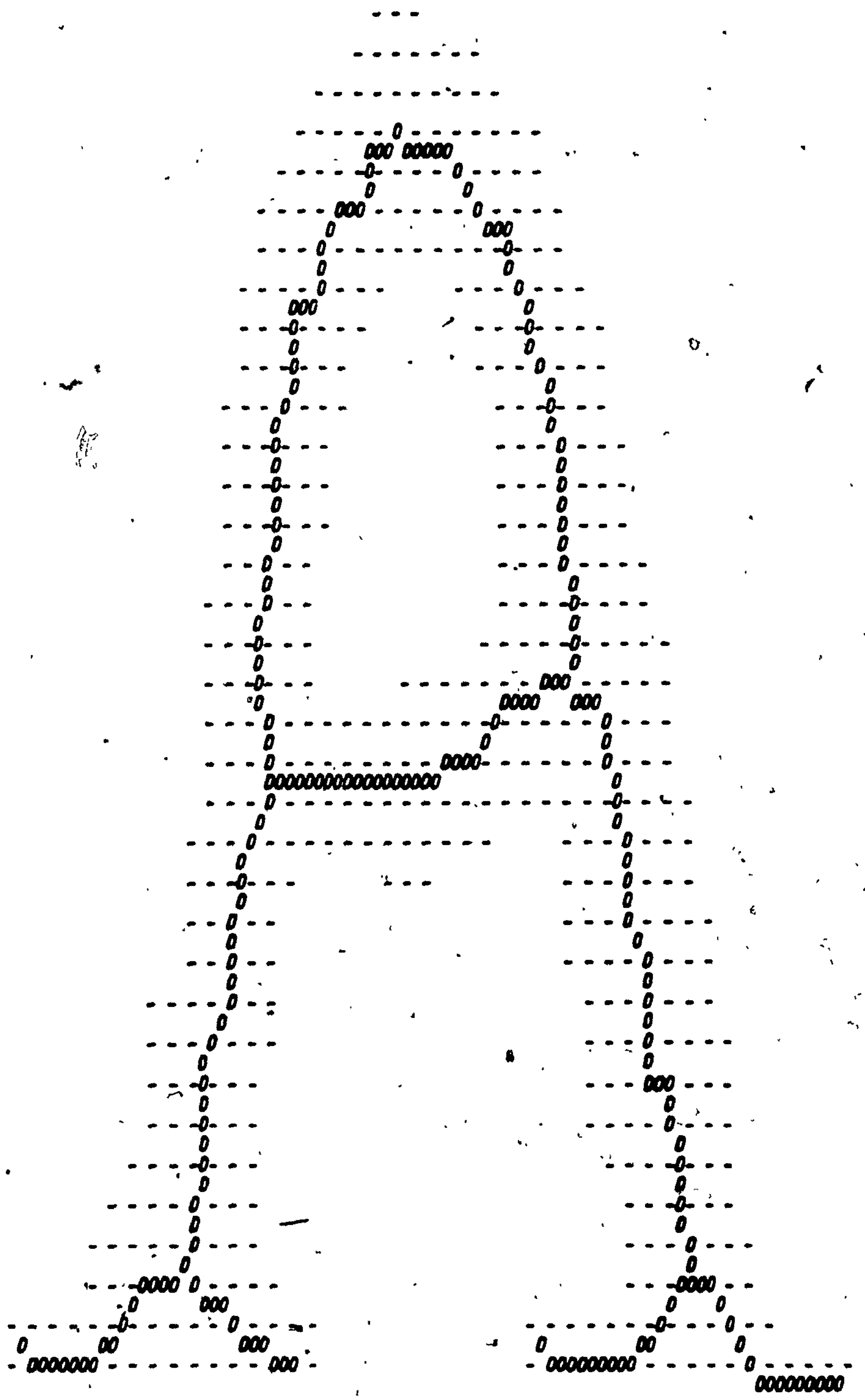
MS2 centre-lines



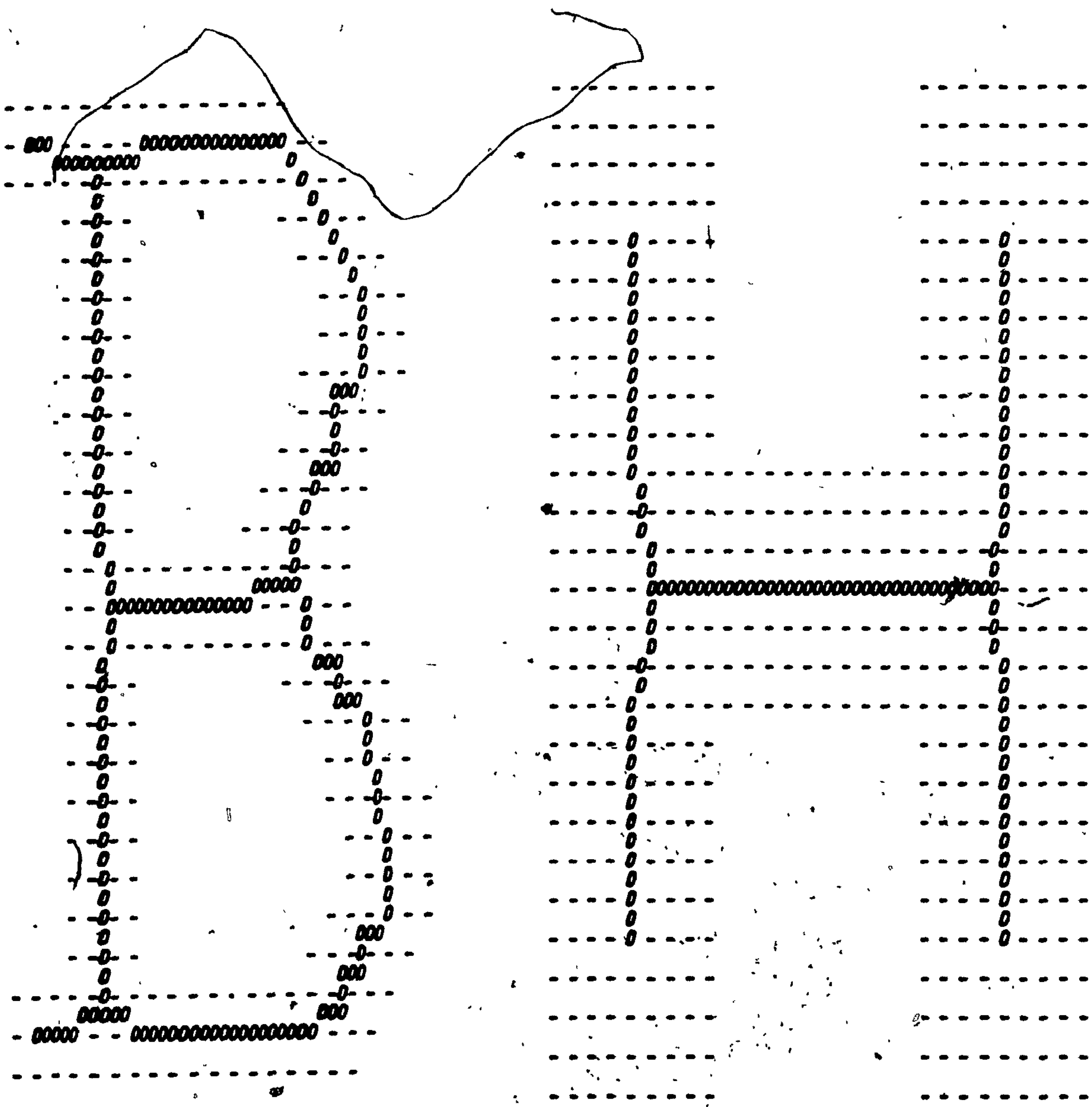
MS2 centre-lines

MS2 centre-lines

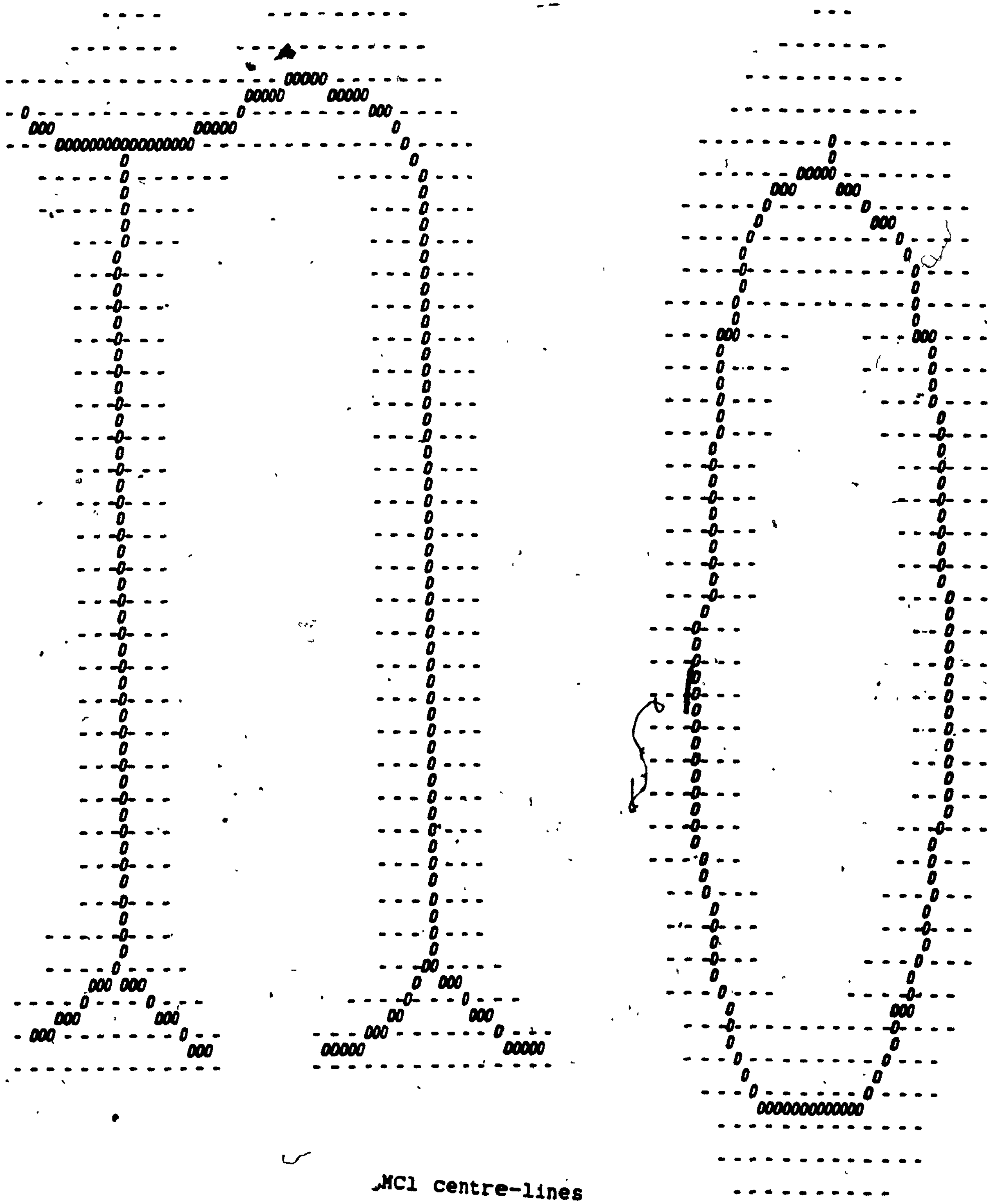
Centre-Lines produced by the MC1 Algorithm



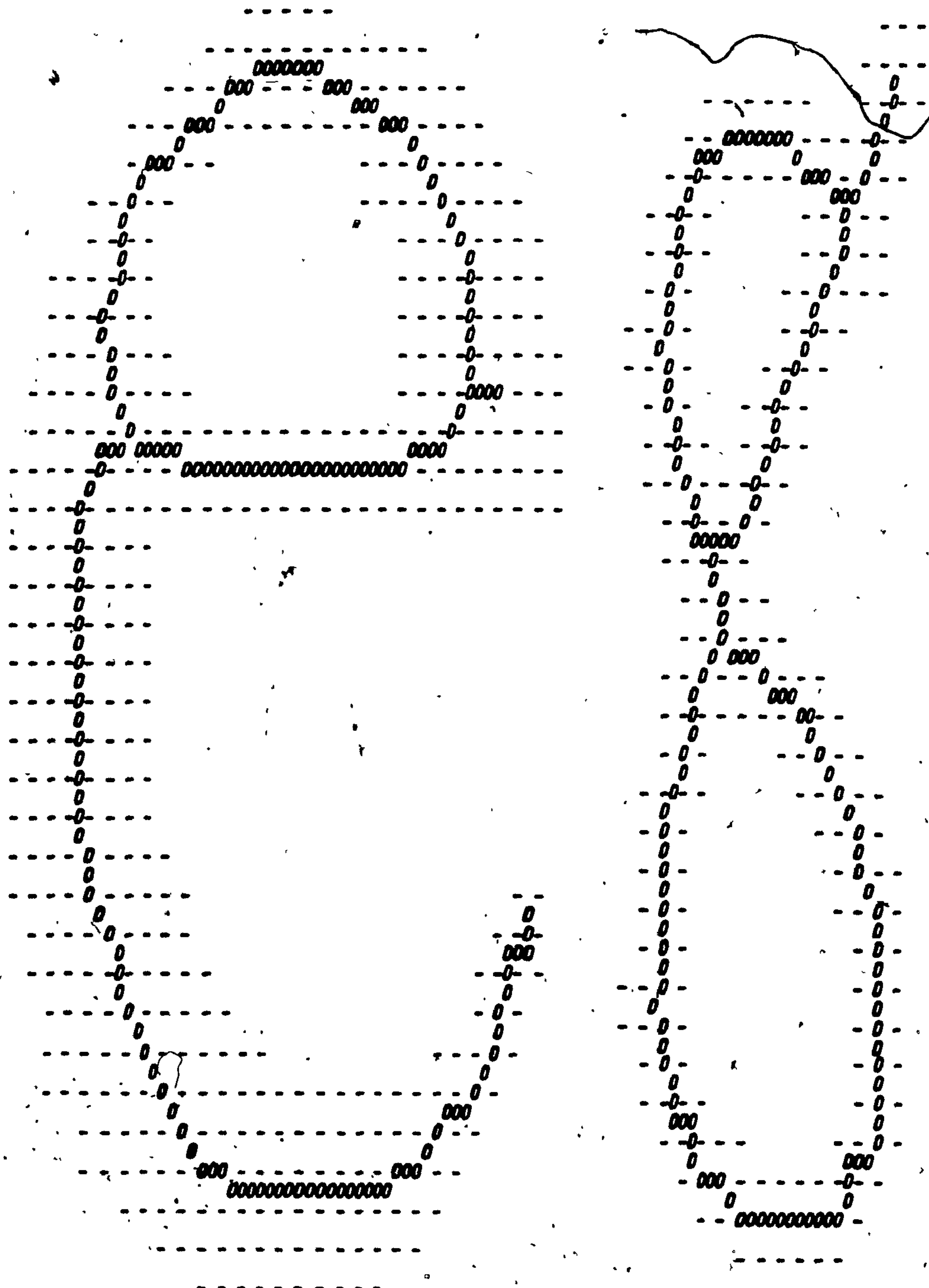
MCl centre-lines



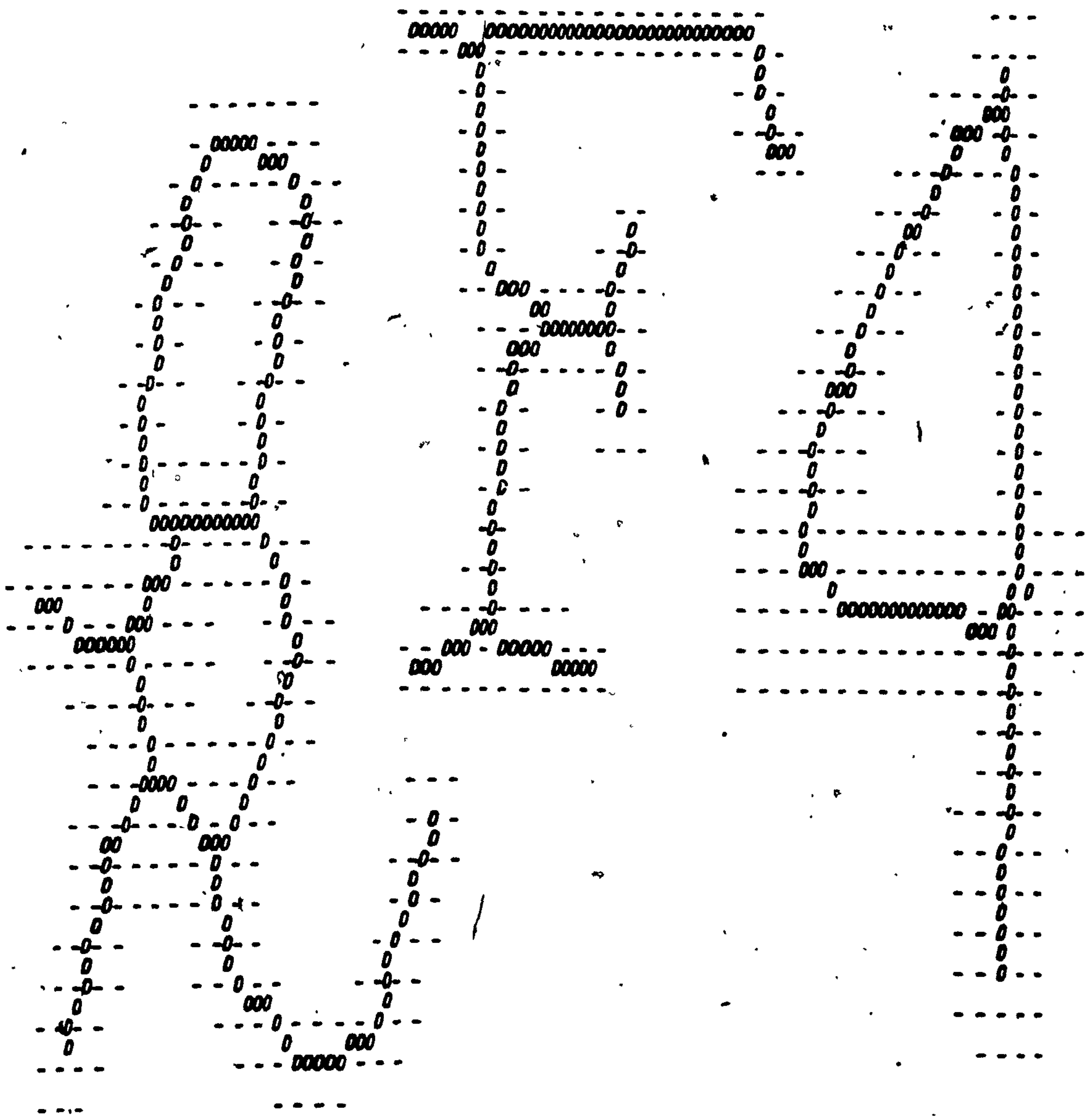
MC1 centre-lines



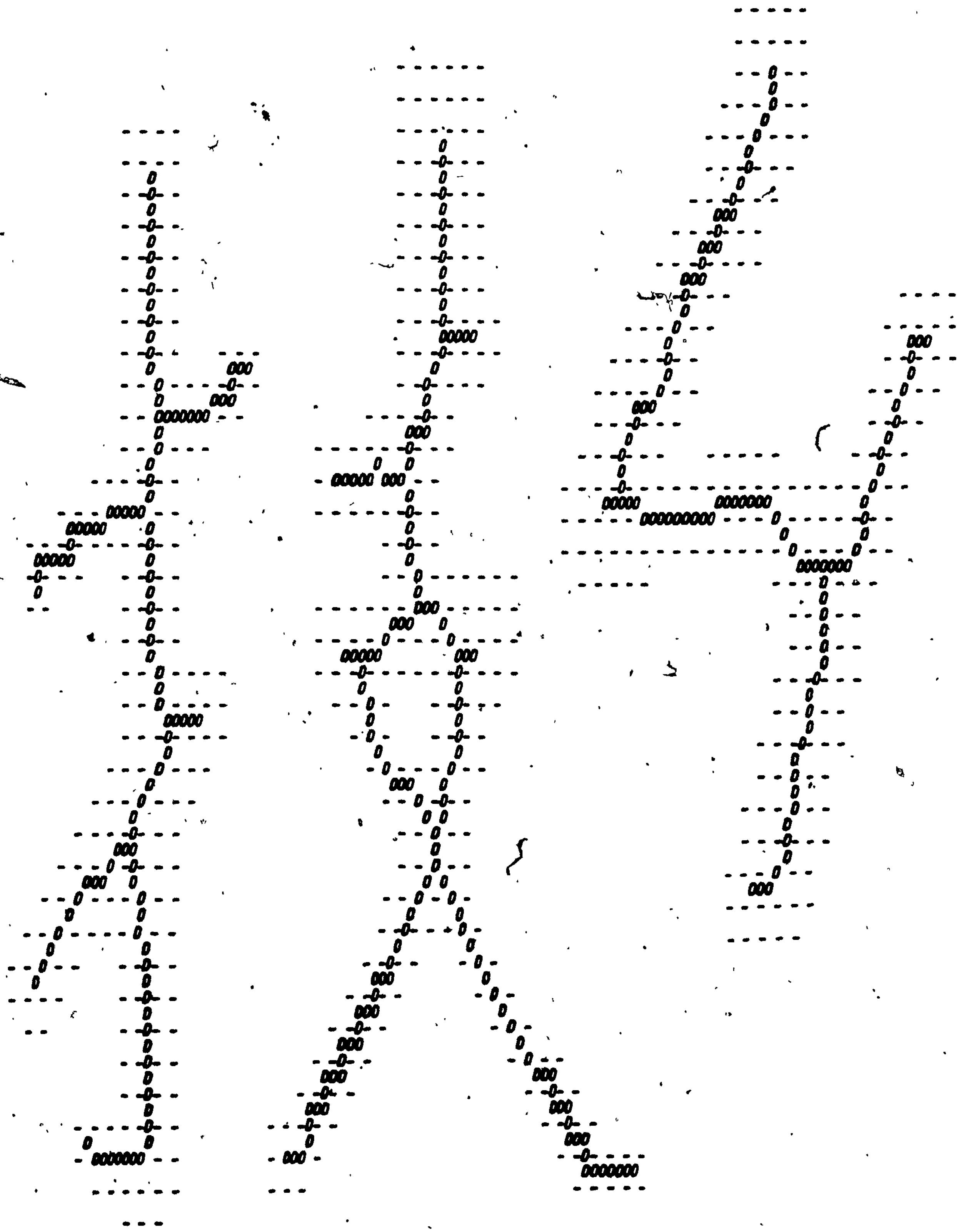
MCl centre-lines



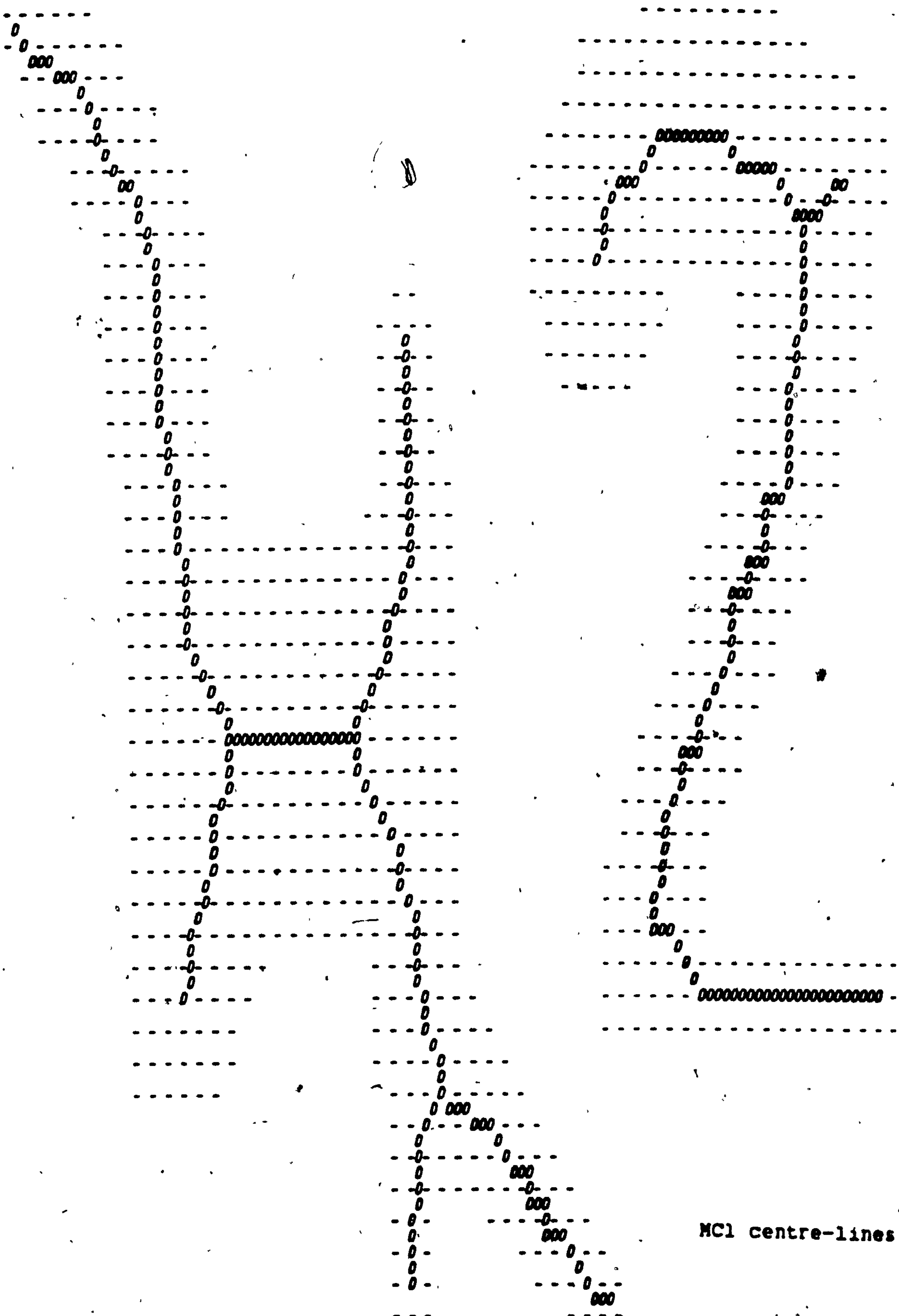
MCl centre-lines



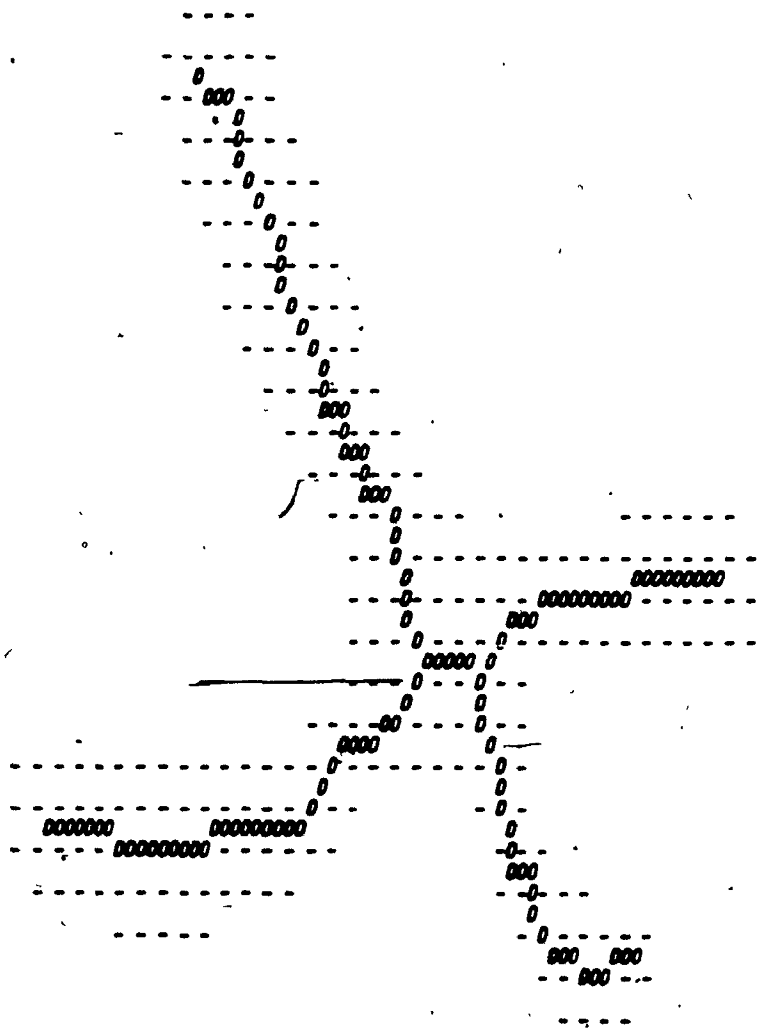
MC1 centre-lines



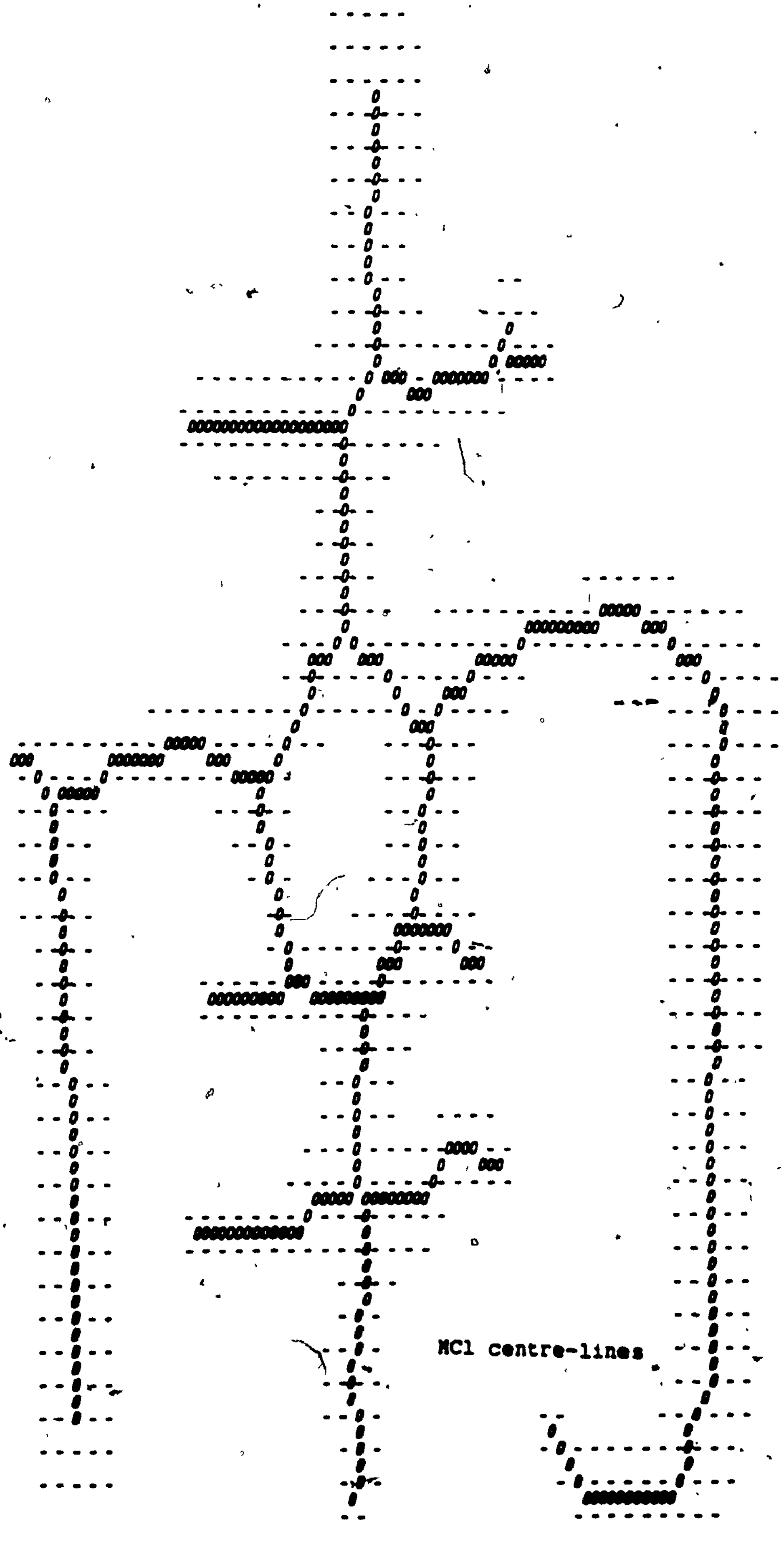
MCl centre-lines

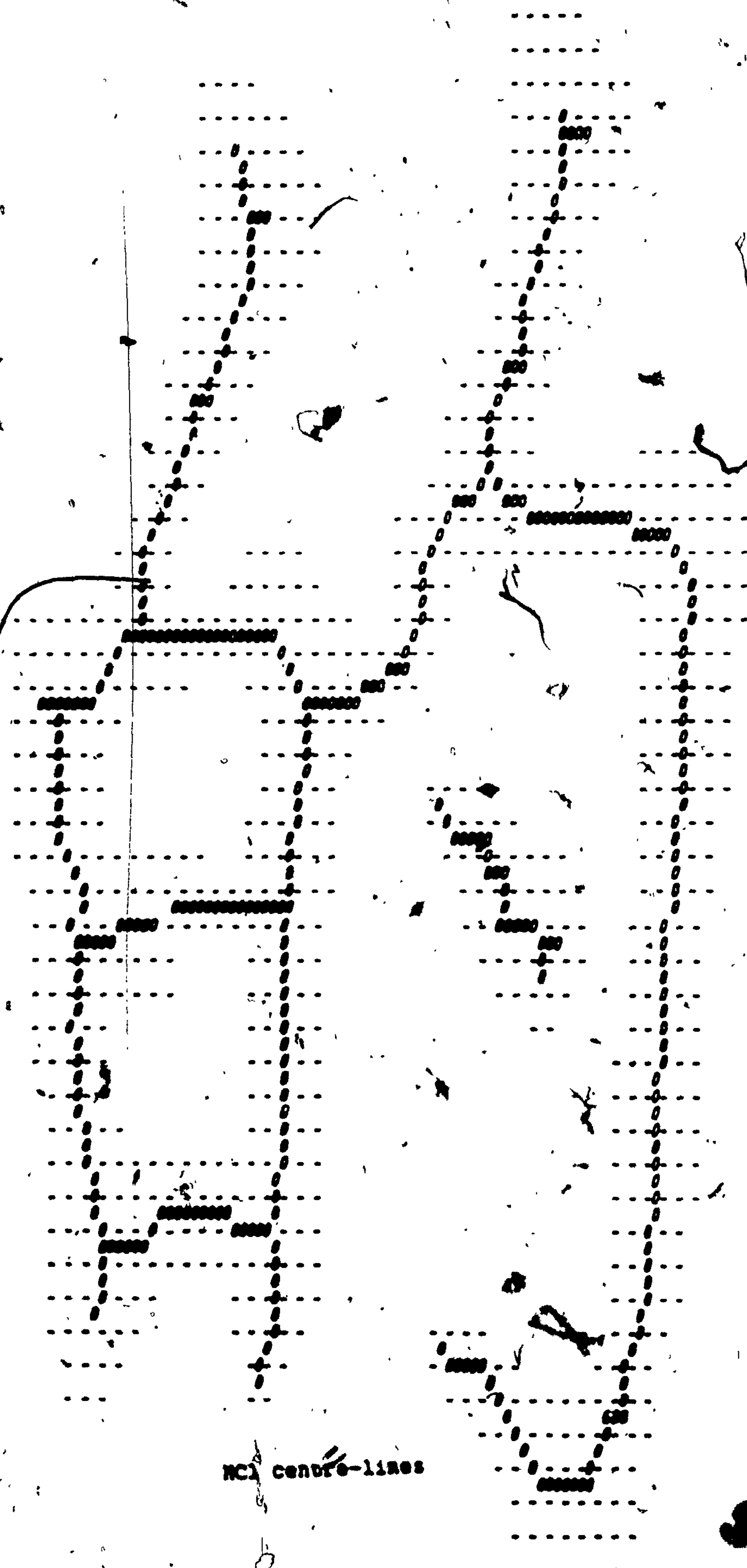


MCl centre-lines

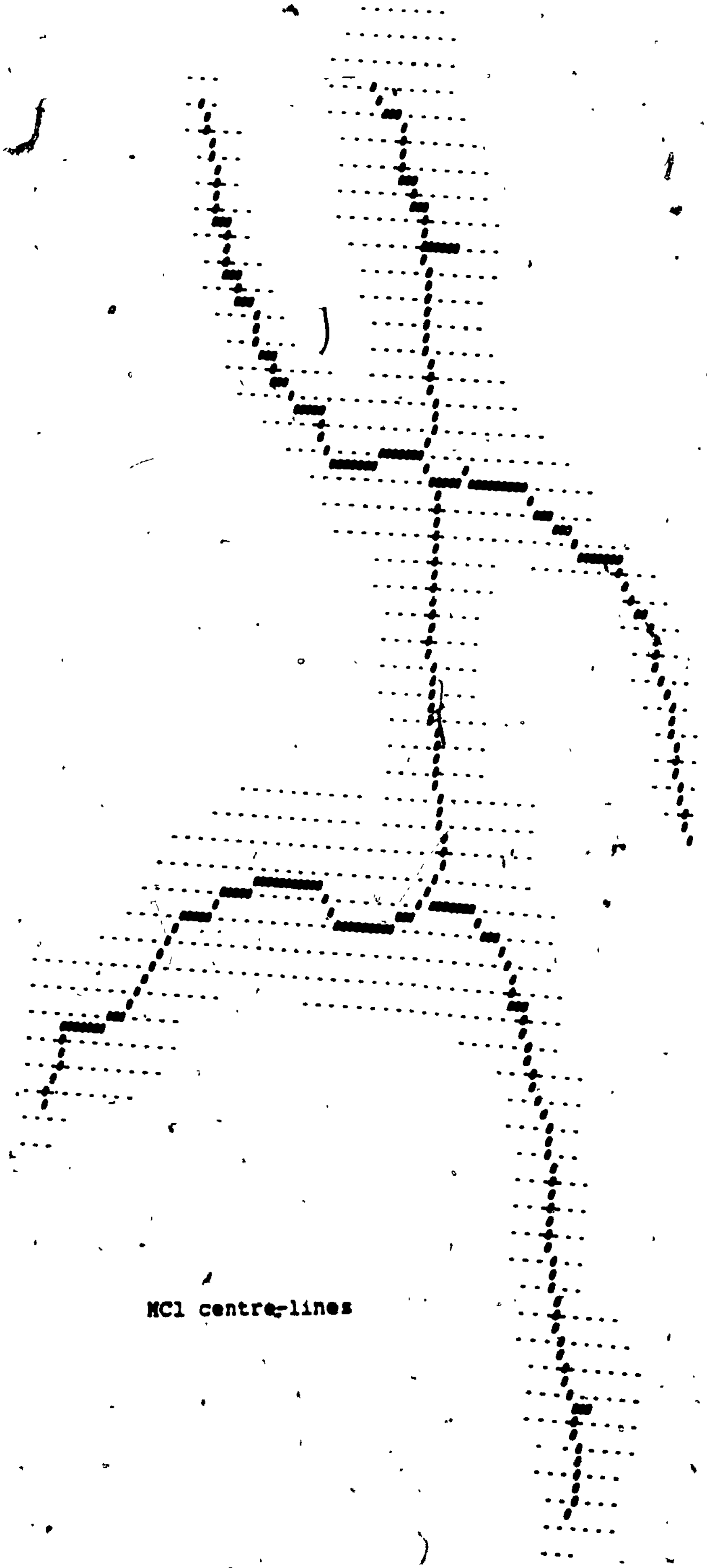


MCl centre-lines


MCl centre-lines

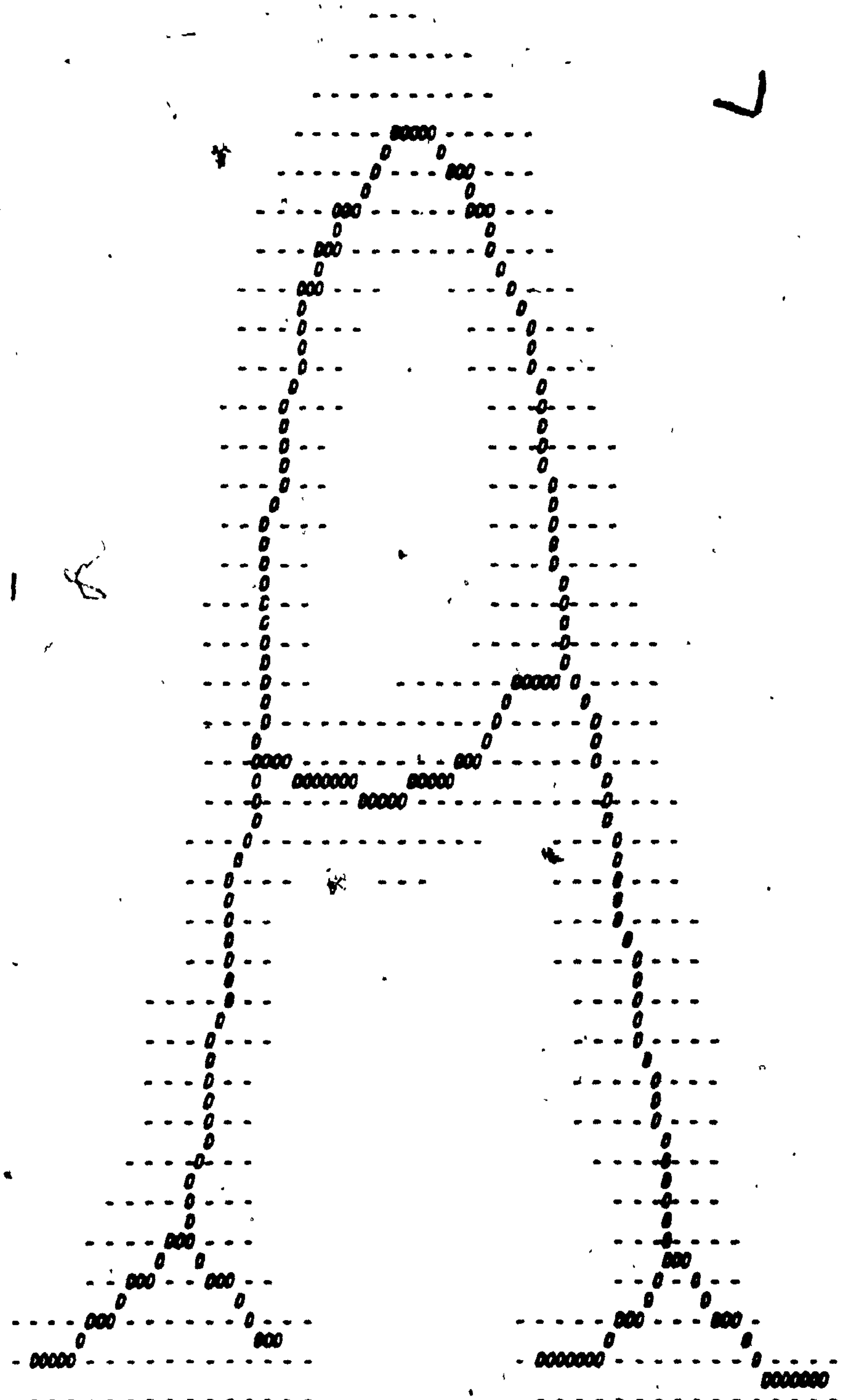


HCl centre-lines

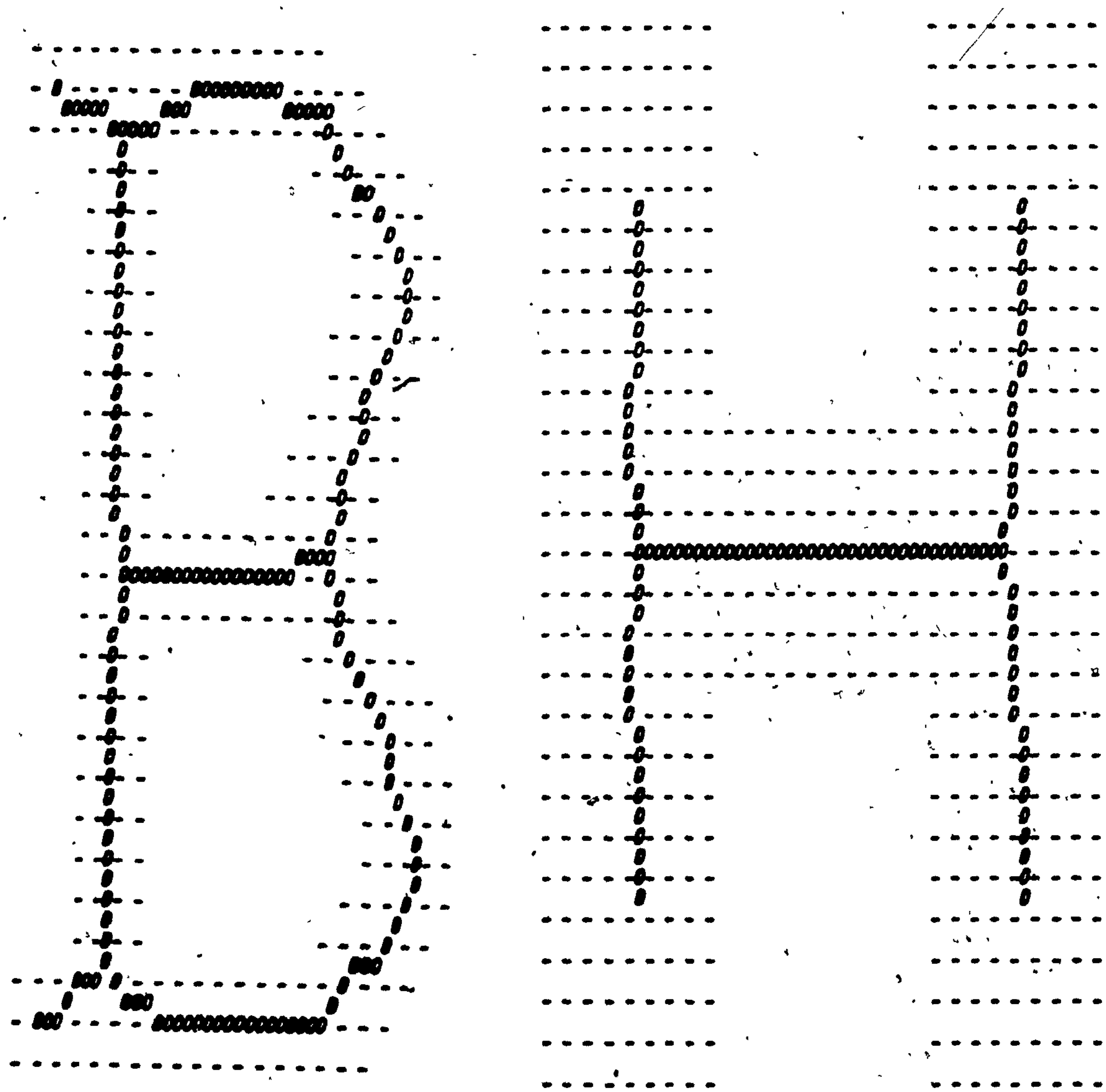


MCl centre-lines

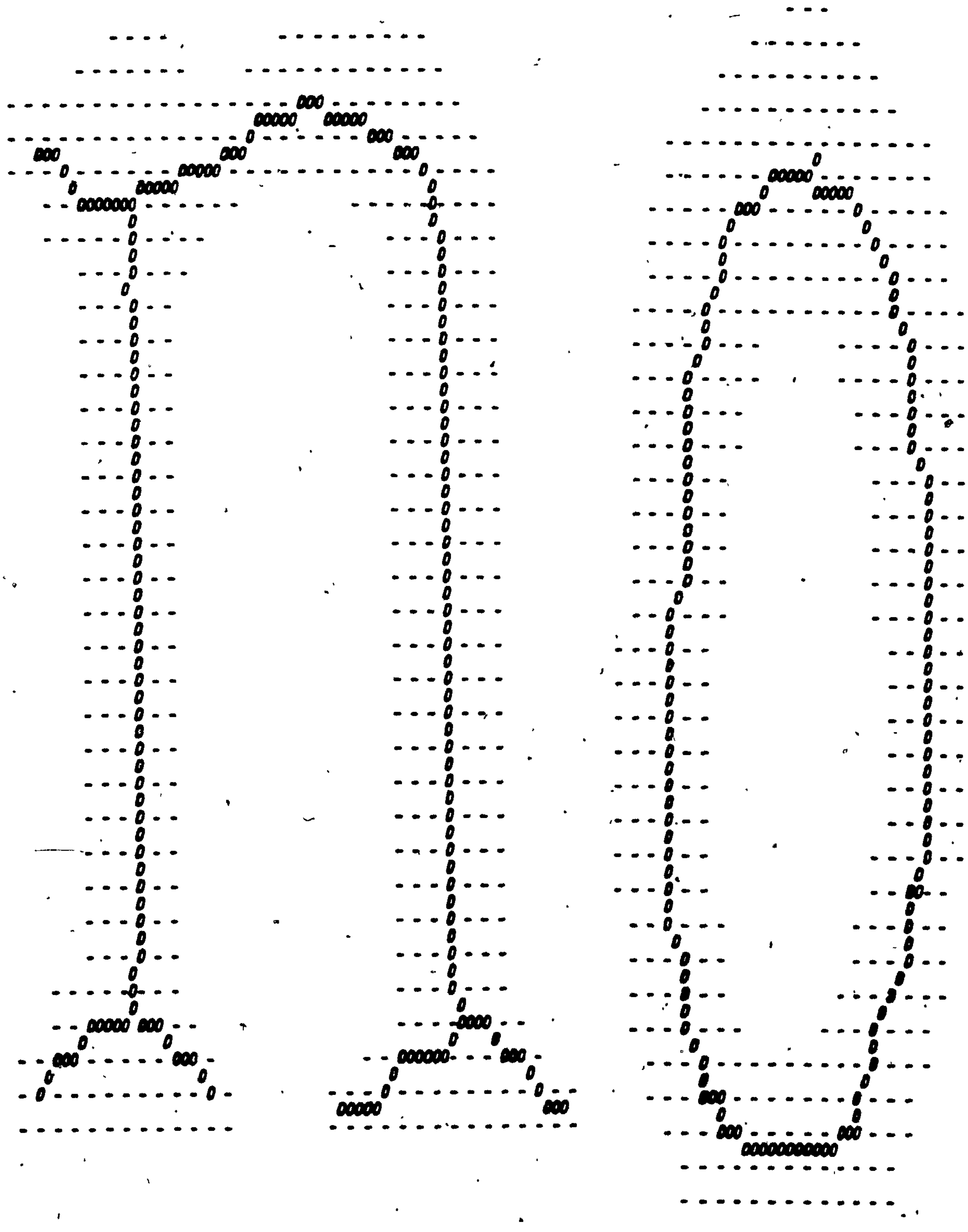
Centre-Lines produced by the MC2 Algorithm



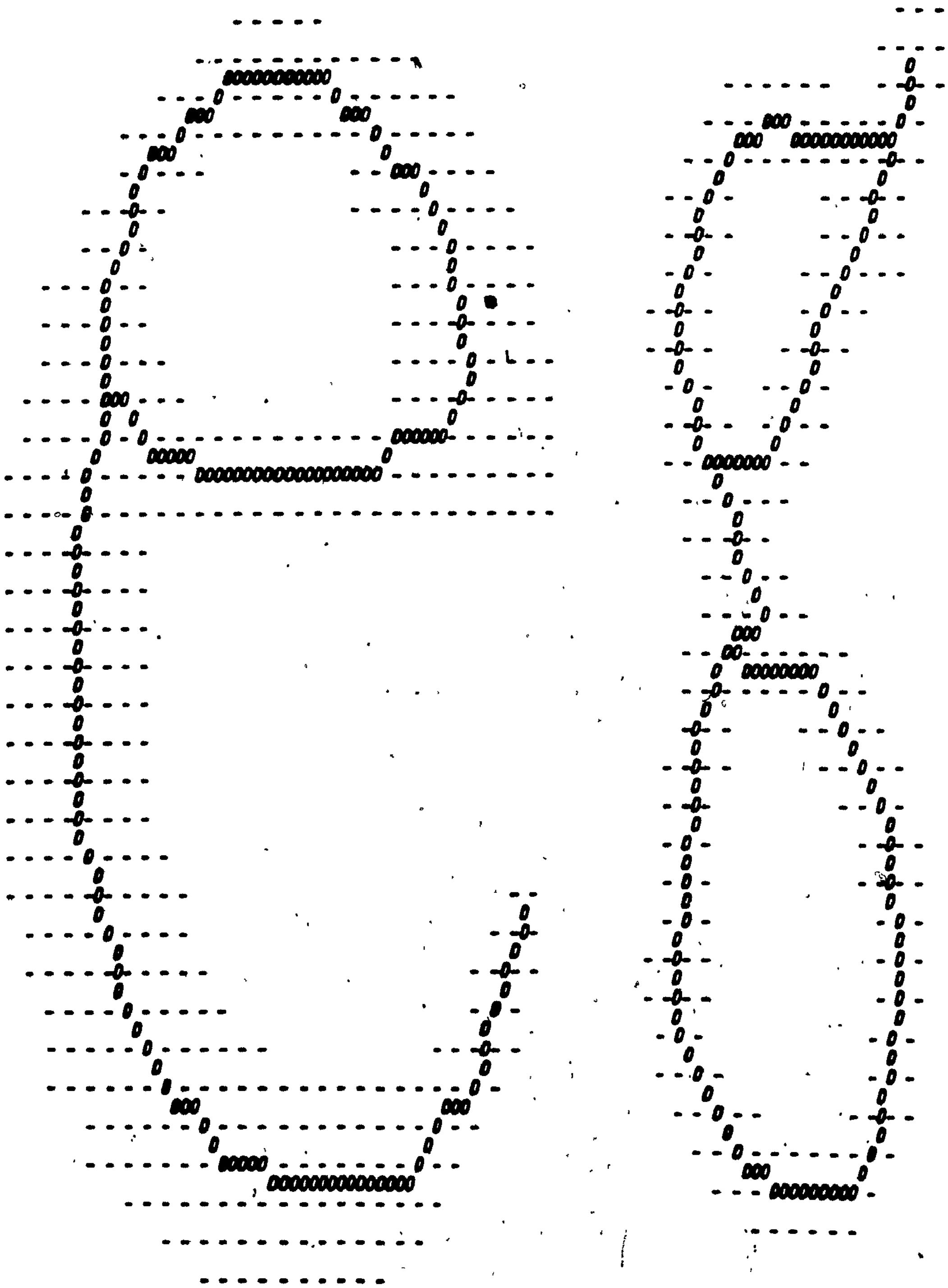
MC2 centre-lines



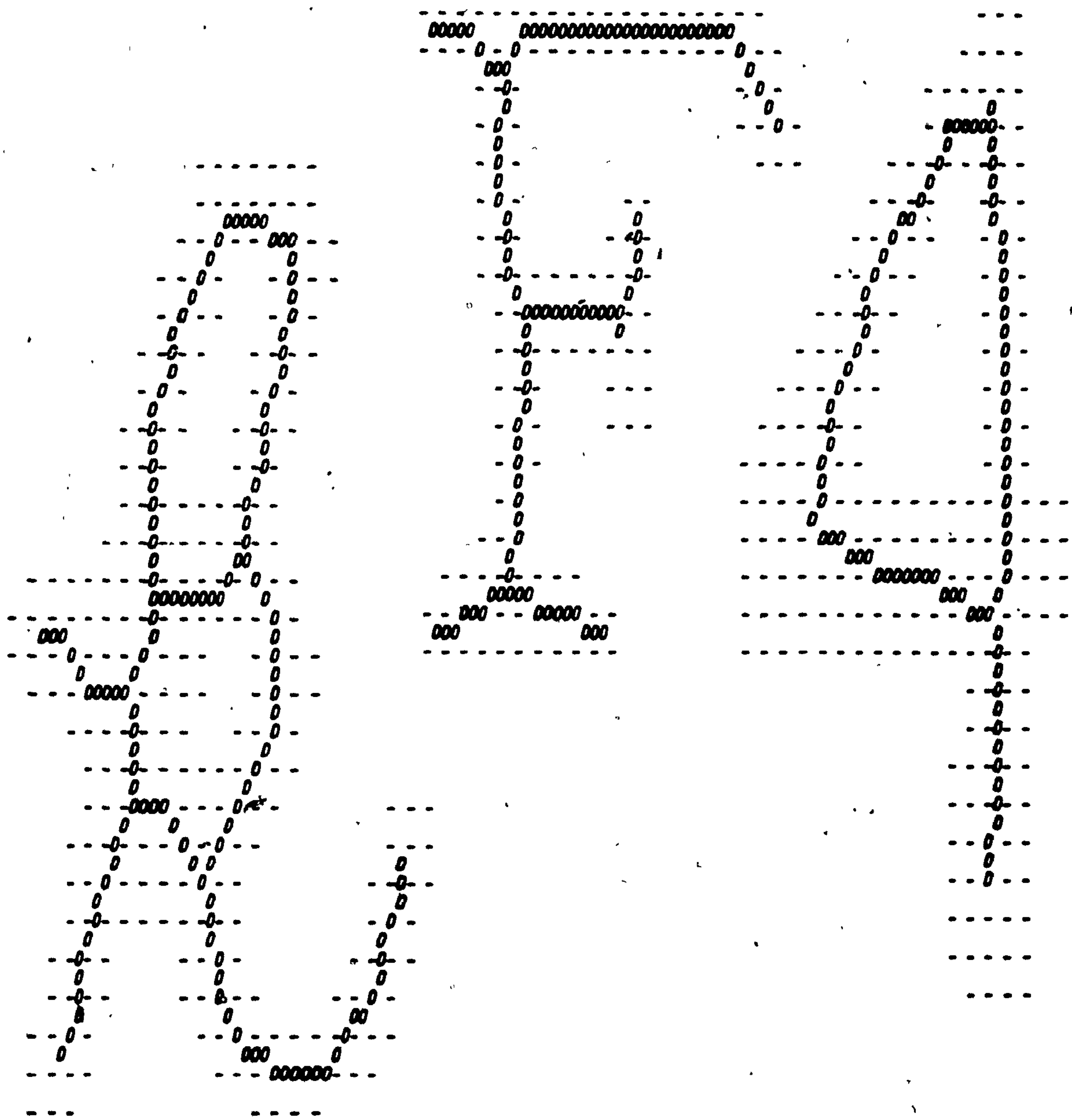
MC2 centre-lines



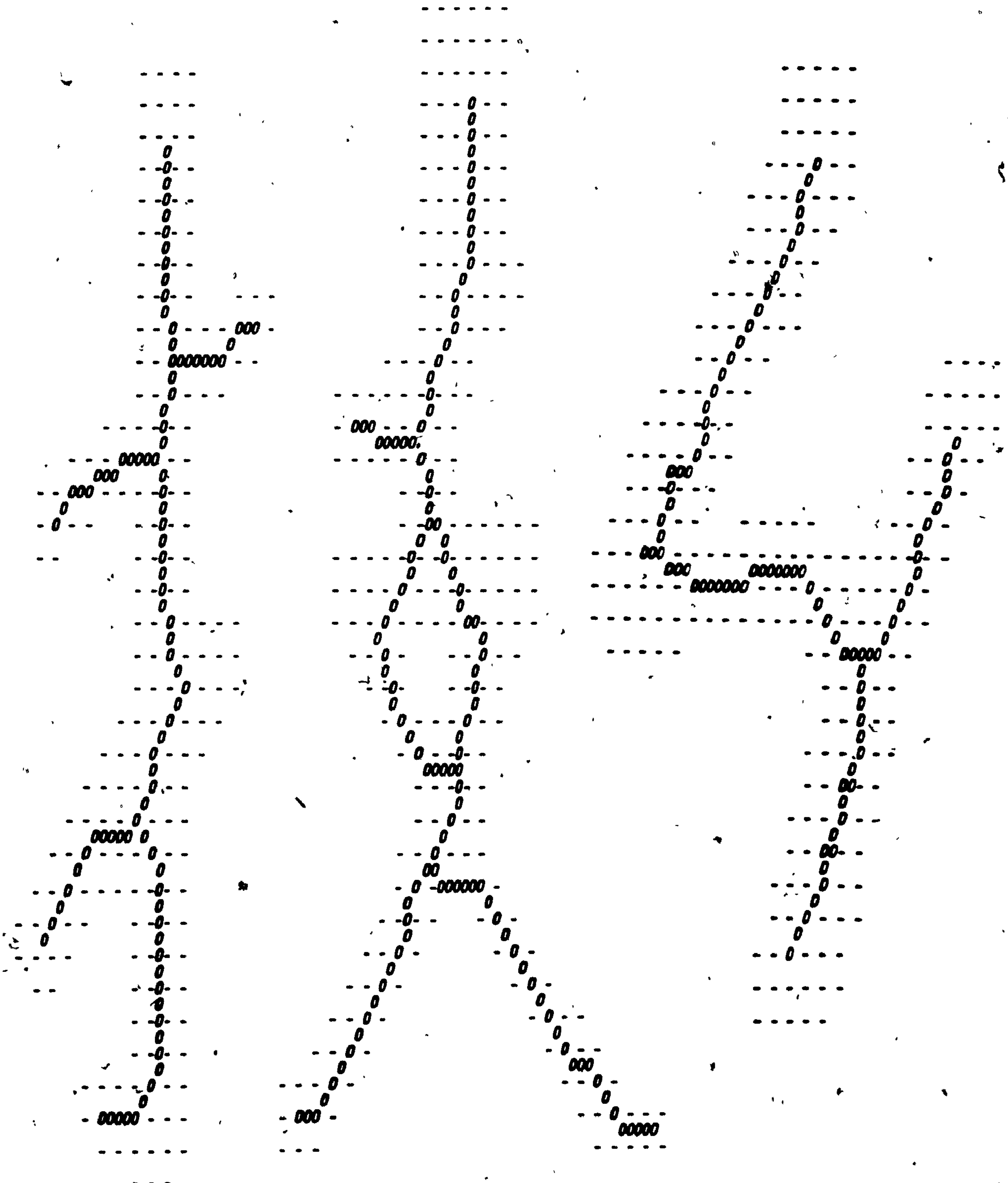
MC2 centre-lines



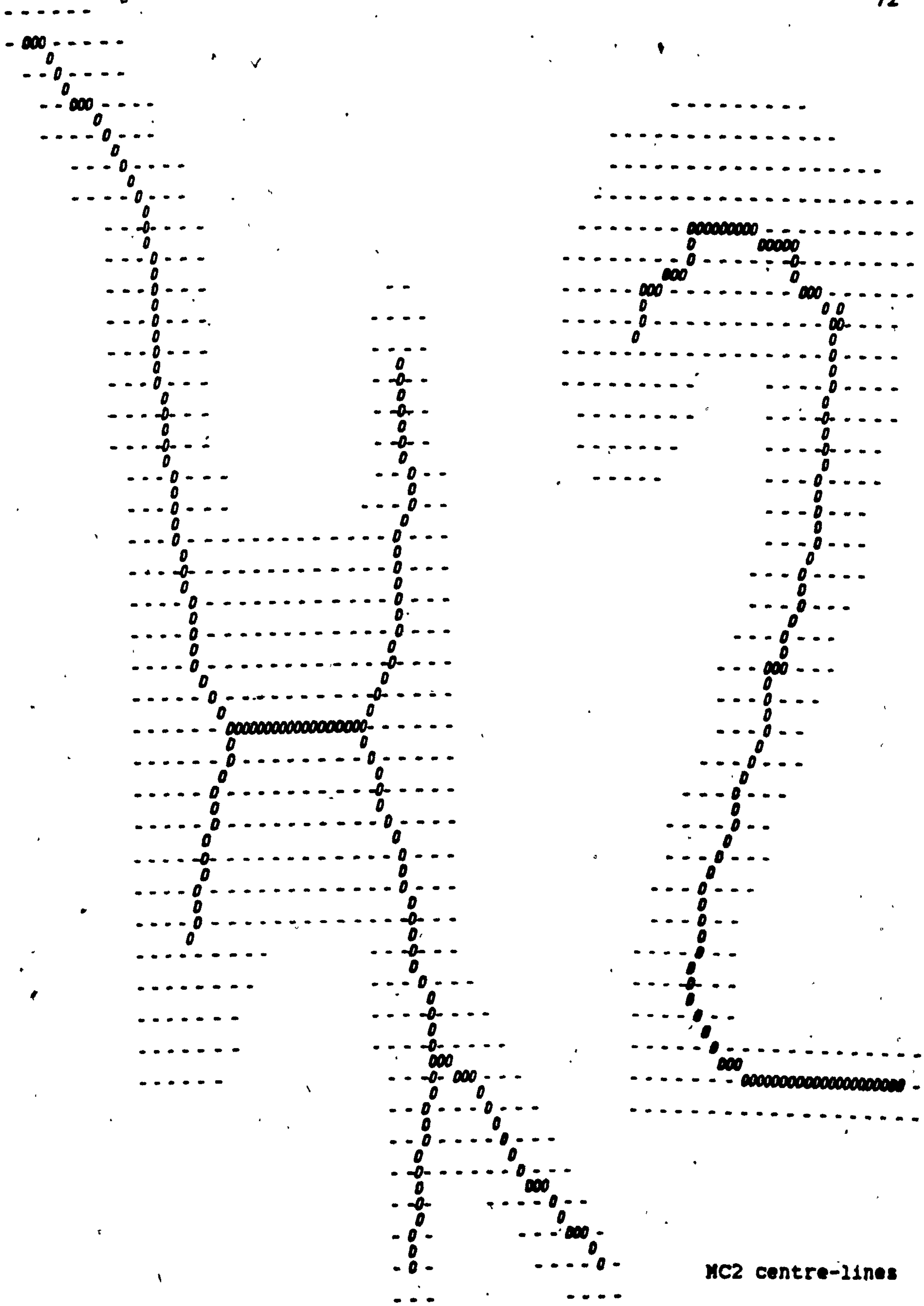
MC2 centre-lines



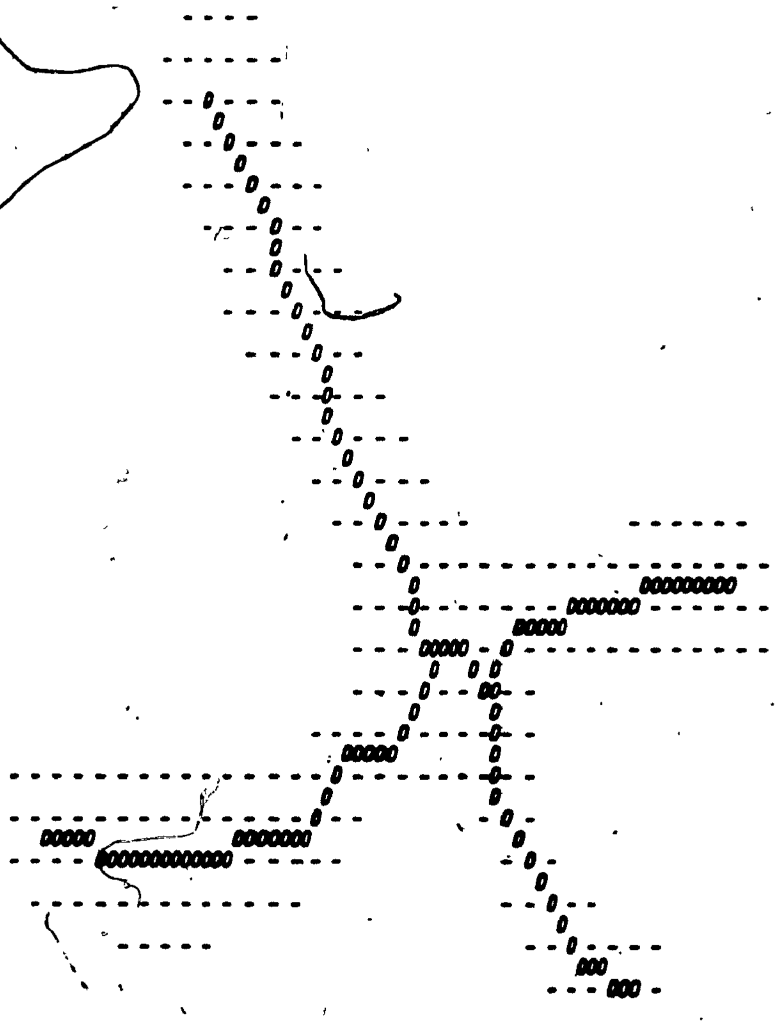
MC2 centre-lines



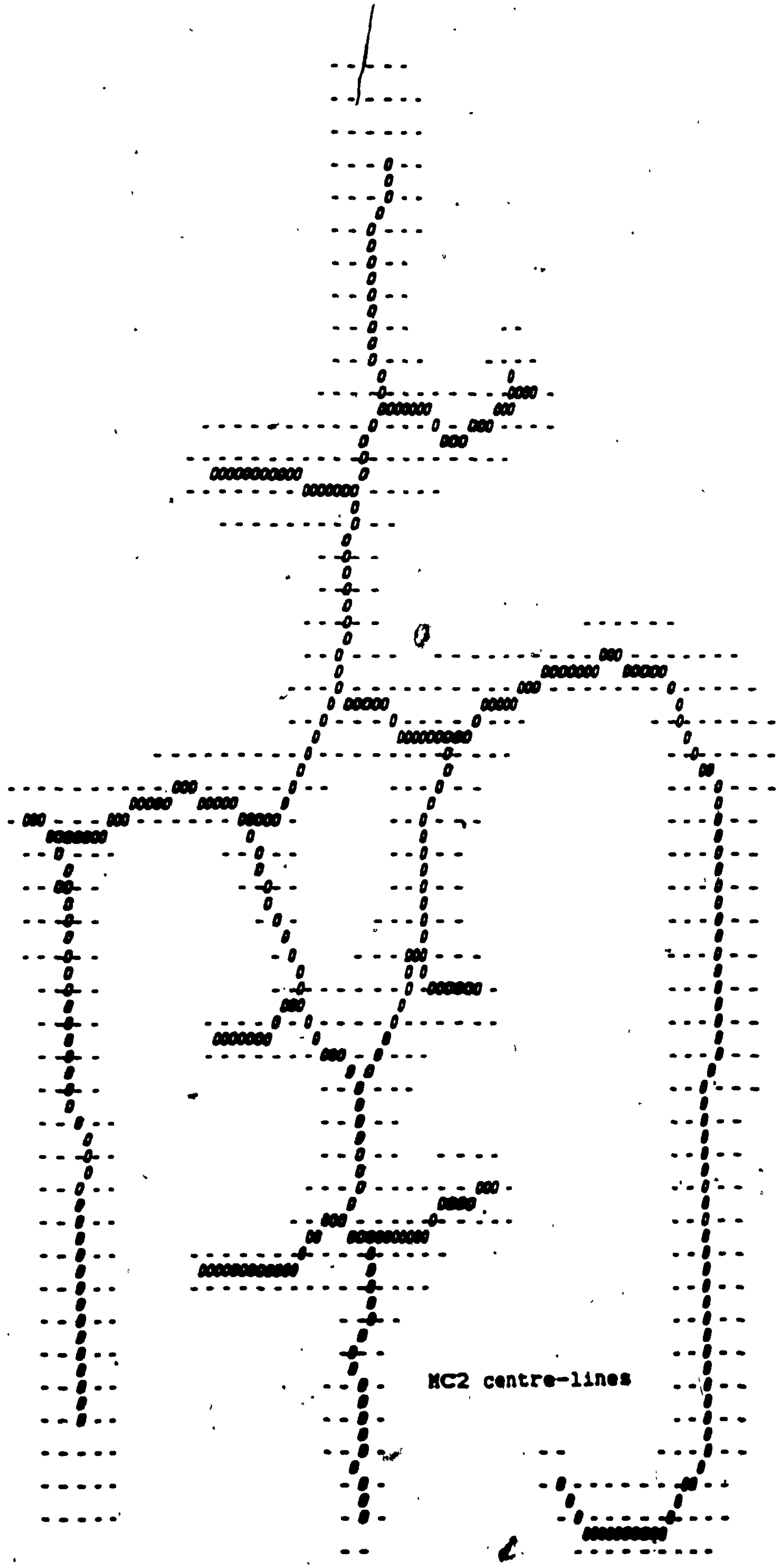
MC2 centre-lines



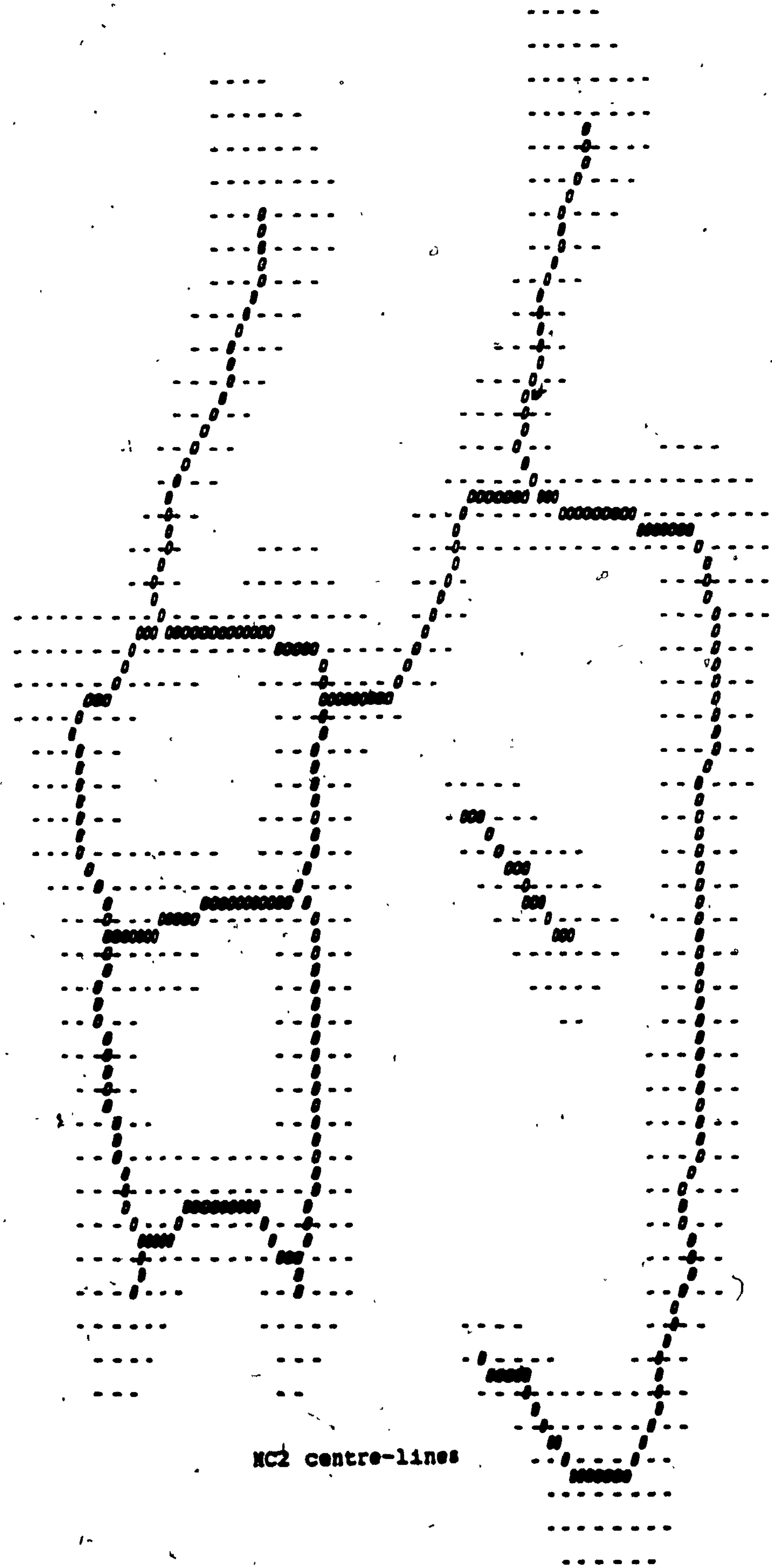
MC2 centre-lines



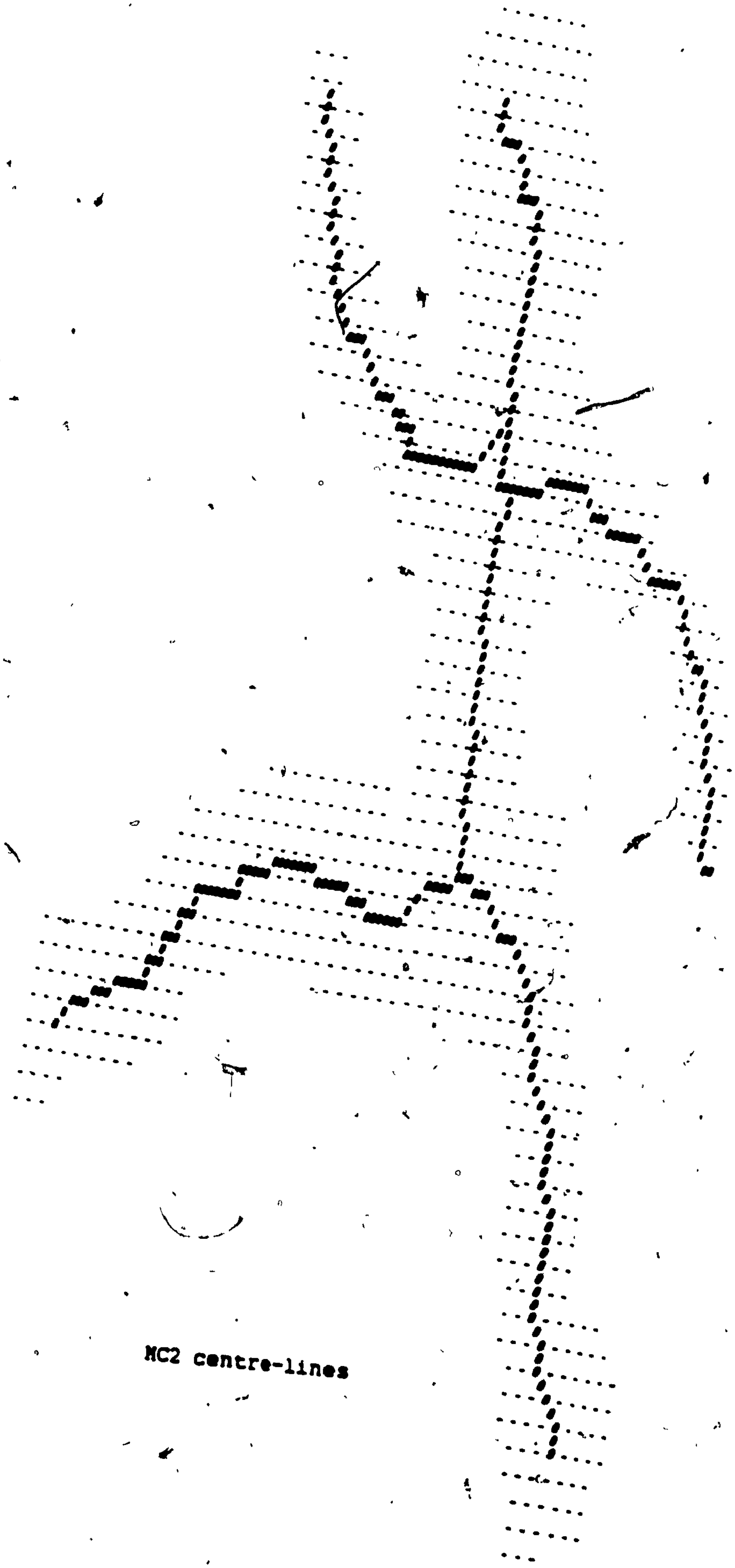
MC2 centre-lines



MC2 centre-lines



MC2 centre-lines



MC2 centre-lines

9 References

[1] Tadao Wakayama, 'A core-line tracing algorithm based on maximal square moving', IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-4, No. 4, No. Jan. 1982.

[2] H. Sherman, 'A quasi-topological method for the recognition of line patterns', Information processing, Proc. UNESCO Conf. pp. 232-238, Butterworths, London 1959.

[3] M. Beun, 'A flexible method for automatic reading of handwritten numerals', Philips Tech. Rev. Vol. 33, pp. 89-101 and 130-137, 1973.

[4] C. V. K. Rao, B. Prasada, and K. R. Sarma, 'An automatic fingerprint calssification system', Proc. 2nd Int. Joint Conf. on pattern recognition, pp. 180-184 1974.

[5] A. P. Pullen, 'Automatic visual inspection of complex industrial components', paper read at BPRA meeting in University college, London, Feb. 1977.

[6] K. J. Udupa and I. S. N. Murthy, 'Some new concepts for encoding line patterns', Pattern Recognition, Vol. 7, pp. 225-233, 1975.

[7] C. J. Hilditch, 'Linear skeletons from square cupboards', Machine Intellingence IV, eds. B. Meltzer and D. Michie, pp. 403-420, Edinburgh Univrsity Press, Edinburgh, 1969.

[8] R. N. Dixon and C. J. Taylor, 'Automated asbestos fibre counting', Proc. IOP Conf. on Machine-Aided Image Analysis pp. 178-185, Institute of Physics, London, 1979.

[9] D. Rutovitz, Pattern Recognition, J. R. stat. soc. 129, pp. 504-530, 1966.

[10] R. M. Jones and M. C. Fairhurst, 'Skeletonigation of binary patterns: a heuristic approach', Electron. Lett. Vol. 14, pp. 265-266 1978.

[11] E. R. Davies and P. N. Plummer, 'Thinning algorithms: A critique and a new methodology', Pattern recognition, Vol. 14, Nos 1-6, pp. 53-63, 1981.

[12] A. Rosenfeld and J. L. Pfaltz, 'Sequential operations in digital picture imaging', Journal of ACM, Vol 13, No. 4, pp. 471-494, 1966.

[13] T. Pavlidis, 'Algorithms for graphics and image processing', Computer Sience Press Inc., Rockville MD, pp. 195-214, 1982.

[14] T. Y. Zhang and C. Y. Suen, 'A fast parallel algorithm for thinning digital patterns', Comm. ACM, Vol. 27, No. 3, pp. 236-239. March 1984.

[15] R. Stefanelli and A. Rosenfield, 'Some parallel algorithms for digital pictures', Journal of the ACM, Vol. 18, No. 2, April 1971.

[16] E. S. Deutsch, 'Thinning algorithms on rectangular, hexagonal, and triangular arrays', Communications of ACM, Vol. 15, No. 9, pp. 827-837, Sep. 1972.

[17] N. J. Naccache and R. Shinghal, 'SPTA - A proposed algorithm for for thinning binary patterns', IEEE Trans. on Systems, Man, and Cybernetics, Vol SMC-14, No. 3, pp. 409-418, May/June 1984.

[18] C. M. Holt, A. Stewart, M. Clint and R. H. Perrott, 'An improved parallel thinning algorithm', Comm. of the ACM Vol. 30, No. 2, pp. 156-160, Feb. 1987.

[19] Yat Keung Chu and C. Y. Suen, 'An alternate smoothing and stripping algorithm for thinning digital binary patterns' Signal Processing, Vol. 11, pp. 207-222, 1986.

[20] H. Ma, 'A comparative study of thinning algorithms in pattern processing', Master's Major Report, Dept. of Computer Science, Concordia Univ., Nov. 1983.

[21] U. Montanari, 'A method for obtaining skeletons using a quasi-Euclidean distance', Journal of the ACM, Vol, 15, pp. 600-624, 1968.

[22] F. L. Bookstein, 'The line skeleton', Computer Graphics and Image Processing, Vol. 11, pp. 123-137, 1979.

[23] C. Arcelli and G. S. de Baja, 'A thinning algorithm based on prominence detection', Pattern Recognition, Vol. 13, No. 3, pp. 225-225, 1981.

[24] S. Zhang and K. S. Fu, 'A thinning algorithm for discrete binary images', 1st International Conf. on Computers and Applications, Beijing, China, pp. 879-886, June 20-22,1984.